Universidad Carlos iii de Madrid
Escuela Politécnica Superior

Departamento de
Teoría de la Señal y Comunicaciones

Universidad
Carlos III de Madrid
www.uc3m.es

Grado en Ingeniería en Tecnologías
de Telecomunicación

Trabajo Fin de Grado

"Detección bioinspirada de saliencia en secuencias de vídeo en Matlab"

Carlos Ruiz Domínguez

Junio de 2014

Universidad Carlos iii de Madrid
Escuela Politécnica Superior

Departamento de
Teoría de la Señal y Comunicaciones

Universidad
Carlos III de Madrid
www.uc3m.es

Grado en Ingeniería en Tecnologías
de Telecomunicación

Trabajo Fin de Grado

"Detección bioinspirada de saliencia en secuencias de vídeo en Matlab"

Autor: Carlos Ruiz Domínguez
Tutora: Carmen Peláez Moreno

Leganés, Junio de 2014

Trabajo Fin de Grado

Detección bioinspirada de saliencia en secuencias de vídeo en Matlab

Autor

Carlos Ruiz Domínguez

Tutora

Carmen Peláez Moreno

La defensa del presente Trabajo Fin de Grado se realizó el día 10 de Julio de 2014, siendo evaluada por el siguiente tribunal:

Presidenta:

Ascensión Gallardo Antolín

Secretario:

Braulio García Cámara

Vocal:

Susana Briz Pacheco

y habiendo obtenido la siguiente Calificación:

Leganés, a 10 de Julio de 2014

Contenido

Agradecimientos

En primer lugar, quiero dar las gracias a mis amigos y mi familia, especialmente a mis padres, por el apoyo recibido durante todos estos años de carrera. A mi padre, que desde niño me guió y me enseñó las bases de todo lo que ahora sé, y gracias al que nunca tuve duda de lo que quería estudiar y trabajar *de mayor*; y por supuesto, a mi madre, que siempre hace absolutamente todo lo que esté en su mano para que pueda centrarme en mis estudios y disfrutar de mis *hobbies* sin tener que preocuparme por otras cosas.

No me olvido de mis compañeros, que me han hecho pasar tantos buenos momentos en los últimos cuatro años. Gracias a ellos he aprendido que estudiar puede ser divertido si se está en buena compañía.

Cómo no, solo tengo palabras de agradecimiento y gratitud hacia mi tutora, Carmen. Durante el último año me ha ayudado mucho, tanto con este proyecto como recomendándome para que el año que viene pueda comenzar a estudiar un doctorado en Estados Unidos. Me ha dedicado buena parte de su tiempo y -sin saberlo- me dio esperanzas en los momentos en que yo pensaba que el proyecto me venía grande y no conseguiríamos buenos resultados a tiempo.

Por último, quisiera dar las gracias a Muratov[1] por cedernos los resultados de algunos de sus experimentos con *eye-tracker*, que nos han servido enormemente como *ground truth* para mejorar nuestro modelo.

Prefacio

Conseguir que una máquina reproduzca o imite comportamientos humanos siempre ha sido un reto complicado.

Durante los últimos años, se han desarrollado varias teorías y numerosas aproximaciones para intentar predecir los objetos o regiones de una fotografía que, por sus características, llamarán la atención del sistema visual humano.

Ahora que estos algoritmos obtienen muy buenos resultados en imágenes, hemos querido ir más allá y proponer nuestro propio modelo para detectar las regiones más llamativas en un vídeo.

Nos hemos dado cuenta de que si utilizamos los algoritmos existentes fotograma a fotograma en secuencias de vídeo, los resultados obtenidos no son los esperados.

Si solo disponemos de una imagen, predicen muy bien las áreas que, por su color, intensidad u orientación nos llaman la atención. Sin embargo, cuando los humanos observamos la realidad, nos fijamos también en muchas cosas por su movimiento -siendo el vuelo de un diminuto mosquito el ejemplo más cotidiano.

Estamos seguros de que las aproximaciones existentes fallan al aplicarse en vídeos porque no tienen en consideración el movimiento de los objetos.

Para comprobar la validez de nuestra hipótesis, vamos a desarrollar un modelo que, basándose en uno de estos algoritmos diseñados para imágenes estáticas, incorpore información adicional como la velocidad o la aceleración para determinar la probabilidad de que, en una secuencia de vídeo, un determinado objeto nos resulte llamativo.

0.1. Entorno socio-económico

A primera vista, puede parecer que conseguir que un ordenador sea capaz de predecir dónde es más probable que se centre la atención de una persona no tiene

ninguna aplicación en el mundo real.

Sin embargo, como veremos detalladamente en la sección 4.2, la utilidad de este tipo de algoritmos y el impacto socio-económico asociado es enorme.

Por ejemplo, sistemas de recursos limitados como pueda ser un robot de exploración en Marte, se podrían beneficiar de esta técnica y centrar toda su capacidad de procesamiento en analizar los datos pertenecientes a los objetos importantes, sin desperdiciar recursos procesando información irrelevante.

Además, nuestro modelo podría ser empleado para mejorar los algoritmos de compresión de vídeo, reduciendo la calidad en las zonas que no sean en absoluto llamativas. Por supuesto, esto tendría un impacto brutal en compañías que almacenan enormes volúmenes de vídeos, como son YouTube, Facebook, Instagram o Vimeo, entre otras.

Para terminar, queremos destacar que en el apéndice B hemos incluido un desglose del presupuesto total para este proyecto.

0.2. Marco regulador

Como ya hemos mencionado, la investigación en modelos de atención espacio-temporal se encuentra actualmente en una fase muy inicial. A estas alturas, no existe ningún marco regulador y es imposible predecir cómo evolucionará este campo en un futuro próximo, así como los usos que se le vayan dando.

Además, en la sección 4.2 veremos la gran variedad de aplicaciones que tiene este sector. En consecuencia, va a ser muy complicado -y quizá innecesario- crear un marco legal genérico que cubra todos los posibles usos.

Probablemente, algunas aplicaciones -como la compresión de vídeos- carecerán de restricciones, mientras que otras tendrán que someterse a determinada legislación. Por ejemplo, si una empresa decidiese utilizar algoritmos de detección de regiones importantes en vídeos procedentes de cámaras de vigilancia, lo más seguro es que tuviesen que certificar la preservación de la privacidad de los individuos que aparezcan en las grabaciones.

Por último, queremos dejar constancia de que todo el código ajeno que hemos empleado para este proyecto tiene licencia abierta para usos educativos y fines de investigación, como es nuestro caso. Además, los datos que utilizamos en el capítulo 3 como *ground truth*, fueron recopilados por Muratov[1] utilizando un *eye-tracker*, previo consentimiento escrito de los individuos que participaron en sus experimentos.

Capítulo 1
Introducción

En este capítulo vamos a explicar a grandes rasgos cómo se encuentra el estado del arte en las diferentes áreas que cubre el presente trabajo.

Comenzaremos hablando de los diversos modelos que existen para determinar las regiones más llamativas para el ojo humano de una imagen (de ahora en adelante, regiones *salientes*, por su término anglosajón *saliency*).

A continuación, haremos un repaso de las diferentes alternativas que se han desarrollado hasta la fecha para tratar de determinar la saliencia de secuencias de vídeo.

Después, estudiaremos las principales técnicas de detección y estimación de movimiento que se emplean en la actualidad, ya que consideramos que el factor del movimiento es la variable necesaria para convertir los modelos de saliencia estática -esto es, aplicables a fotografías- en modelos válidos para secuencias de vídeo.

También comentaremos el problema de la segmentación de imágenes y las distintas soluciones propuestas hasta ahora, puesto que para nuestro modelo requeriremos agrupar píxeles con características similares en regiones o *segmentos*.

Por último, concluiremos la introducción con un breve resumen de algunas de las técnicas de aprendizaje máquina más extendidas hoy en día, para explicar cómo hemos entrenado a una red neuronal para estimar la saliencia de un vídeo.

1.1. Saliencia espacial

Desde principios de la década de los 90 la comunidad científica especializada en visión artificial ha tratado de abordar el problema de la detección de las regiones

más salientes. Si bien algunos autores han intentado estudiar cómo funciona el sistema visual en primates y humanos desde un enfoque *neurobiológico* -analizando la actividad cerebral de varios individuos al observar diversas fotografías[2]- la mayoría han optado por proponer aproximaciones basadas en modelados *estadísticos*.

Pero antes de hablar de los principales algoritmos desarrollados durante estos últimos años, nos gustaría comenzar con un par de anotaciones. En primer lugar, cabe destacar que más allá del puro interés científico por entender cómo funciona nuestro sistema de atención visual, la habilidad de predecir las regiones en las que un humano se puede fijar tiene enormes ventajas, como se discutirá extensamente en la sección correspondiente del capítulo de conclusiones (sec. 4.2).

En segundo lugar, creemos oportuno establecer una definición formal de lo que se conoce como un mapa de saliencia, que no es más que una representación topográfica de la relevancia e intensidad relativa de los estímulos en el espacio visual[3]. Por tanto, un mapa de saliencia representa las zonas potenciales, ordenadas siguiendo un determinado criterio, a las que más probablemente se desplazarán los ojos de una persona.

Dicho esto, ya podemos empezar con el modelo de saliencia espacial más citado hasta ahora -y en el que nos hemos basado en este proyecto para extenderlo a vídeos-, que es el de Itti et al.[4]. Aunque entraremos en un mayor nivel de detalle en el siguiente capítulo, hemos resumido a continuación los cuatro pasos principales del modelo, como se observa en la figura 1.1:

1. **Extracción de características:** Aplicando sencillas técnicas de filtrado, el algoritmo recopila información de las variaciones en color, intensidad y orientación de cada uno de los píxeles de la imagen original, gracias al cálculo de las diferencias *center-surround* entre distintos niveles de una *pirámide gaussiana* -en cada nivel, la imagen original se diezma por un factor distinto-.

2. **Obtención de los mapas de características:** Una vez calculadas las diferencias *center-surround* en todos los niveles de la pirámide, el siguiente paso combina la información de todos los niveles en un único mapa para cada característica (color, intensidad y orientación).

3. **Obtención del mapa de saliencia:** Utilizando una combinación lineal de los tres mapas de características, se obtiene el mapa final de saliencia.

4. **Detección de la región saliente:** Siguiendo la técnica neuronal del *winner-takes-all*, se detecta la región saliente donde el estímulo del mapa de saliencia haya sido más relevante.

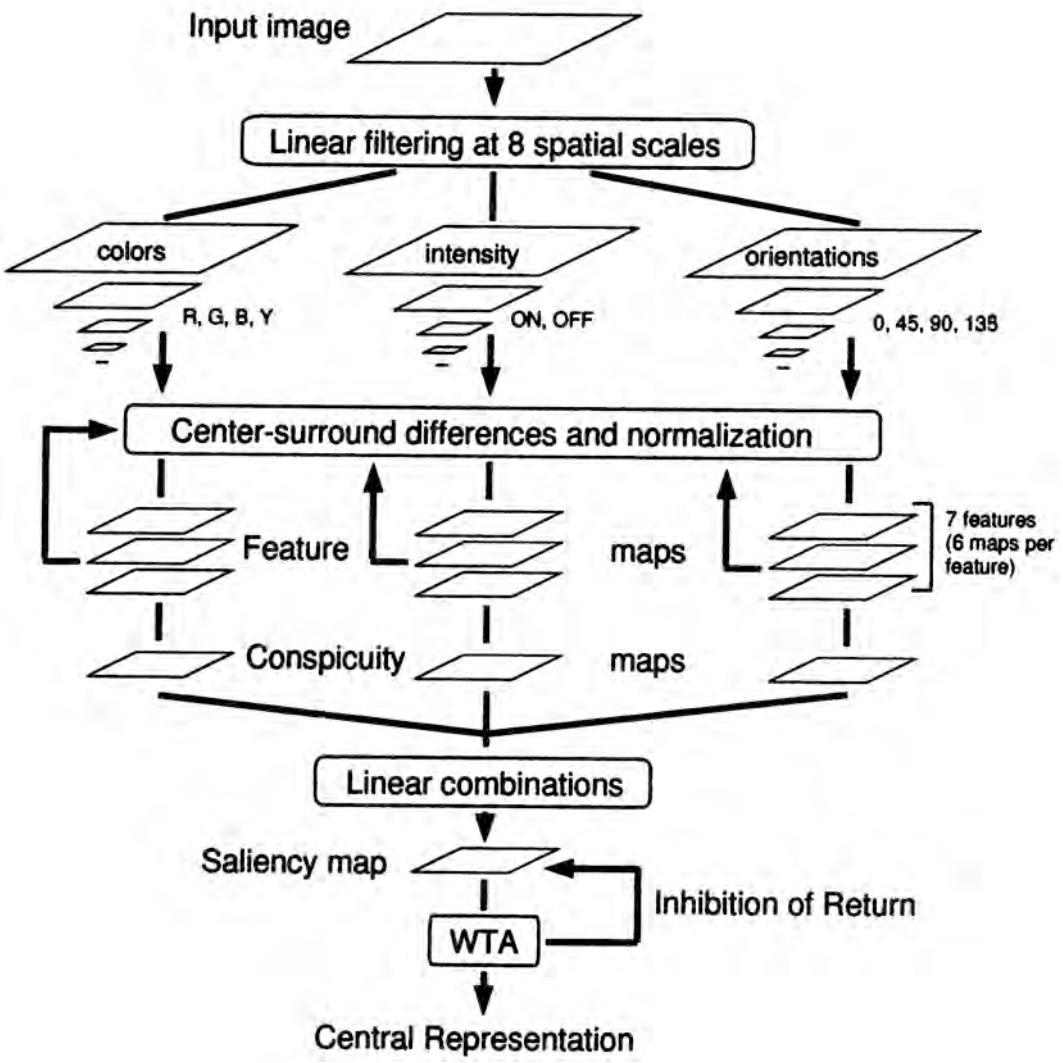


Figura 1.1. Modelo de saliencia espacial de Itti et al.[4].

Posteriormente, los autores del modelo añadieron un algoritmo de normalización a la salida del paso 3 -obtención del mapa de saliencia- gracias al cual se podía predecir no sólo la región más saliente sino las siguientes zonas más llamativas, a las que probablemente se desplazaría la atención visual después de centrarse en la región ganadora[5].

Desde que Itti y Koch publicasen su mejorado modelo en el año 2000, se han sucedido numerosas propuestas y enfoques distintos para tratar de superar su efectividad. Así, por ejemplo, Bruce y Tsotsos[6] propusieron un método que mejora la eficacia del algoritmo maximizando la información extraída de las medidas tomadas de la escena. Por su parte, Schölkopf et al.[7] mejoraron la eficiencia del proceso al beneficiarse de la estructura topográfica y el paralelismo de los algoritmos de grafos, definiendo cadenas de Markov sobre varios mapas de características.

Recientemente los investigadores han ido probando distintas estrategias para conseguir mejores resultados. Por ejemplo, Hou y Zhang [8] propusieron un modelo que no se basa en extraer características ni en utilizar conocimiento previo sobre los objetos de una escena. En su lugar, proponen un método rápido para calcular la saliencia analizando el *log-espectro* (*log*(intensidad) vs. *log*(frecuencia)) de la imagen.

También Judd et al.[9] consiguieron excelentes resultados entrenando su modelo con datos de un *eye-tracker* (dispositivo capaz de detectar dónde mira una persona). Para ello, de cada imagen escogieron diez píxeles al azar de entre el 20 % más saliente y otros diez del 70 % menos relevante. Sus resultados son bastante precisos, pero dependientes del tipo de imágenes sobre el que fue entrenado el modelo, que por ejemplo utiliza el algoritmo de detección de rostros de Viola y Jones [10].

Por último, debemos destacar la idea desarrollada por Perazzi et al.[11], que ha conseguido mejorar con creces los resultados de los modelos anteriores con su algoritmo. Como podemos ver en la figura 1.2, se siguen tres pasos para determinar el mapa de saliencia. Inicialmente, se abstrae la imagen original, agrupando píxeles con colores parecidos en pequeñas regiones homogéneas. En segundo lugar, se calculan dos mapas de características: el de singularidad (comparación de cada segmento con el color y la posición del resto de segmentos) y el de distribución (en función de la varianza espacial del color de cada segmento). Finalmente, se asigna un valor de saliencia a cada píxel basándose en una combinación lineal de las saliencias de los segmentos de alrededor.

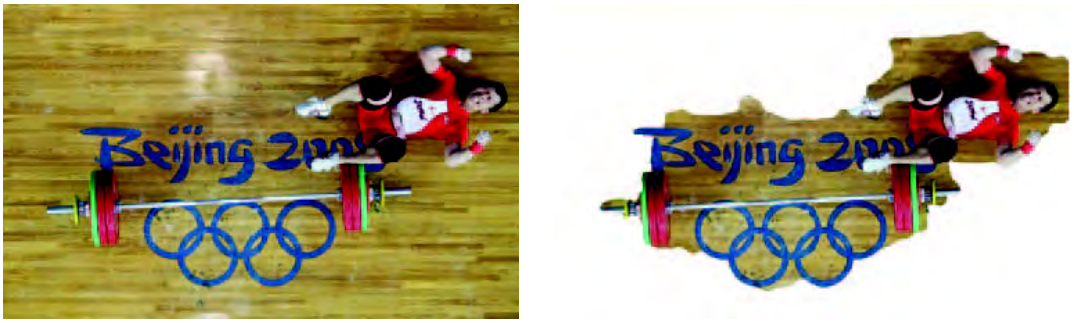
(a) Imagen original | **(b)** Abstracción | **(c)** Singularidad | **(d)** Distribución | **(e)** Mapa de saliencia | **(f)** *Ground truth*

Figura 1.2. Revolucionario modelo de saliencia propuesto por Perazzi et al.[11]. Esta figura muestra un ejemplo de los tres pasos del algoritmo: abstracción (b), obtención de los mapas de características (c y d) y elaboración del mapa final de saliencia (e).

Además, en la figura 1.3 hemos añadido varios ejemplos que demuestran los espléndidos resultados logrados por el modelo de Perazzi et al.[11]. Como nota, cabe mencionar que Cheng et al.[12] añadieron a finales del año pasado una pequeña mejora en la eficiencia de algunos pasos del algoritmo, consiguiendo reducir el tiempo medio de ejecución de 150ms a 90ms -para imágenes de resolución 300x400.

Figura 1.3. Ejemplos de los mapas de saliencia obtenidos por Perazzi et al.[11].
De izquierda a derecha: imagen original, abstracción, mapa de saliencia y *ground truth*.

Aunque como ya hemos mencionado anteriormente hablaremos más en profundidad de las aplicaciones reales de la detección de saliencia en la sección correspondiente del capítulo de conclusiones (sec. 4.2), nos ha parecido interesante incluir en la figura 1.4 un ejemplo actual del uso de la(s) región(es) saliente(s) para reconocer y describir el contexto de la escena, como propusieron Goferman et al.[13].



(a) Imagen original **(b)** Región saliente

Figura 1.4. Nuevo uso de la saliencia espacial descrito por Goferman et al.[13].
Las regiones salientes pueden servir para reconocer y describir el contexto de la escena.

1.2. Saliencia dinámica

La investigación en la detección de saliencia dinámica o *espacio-temporal* no comenzó a dar sus frutos hasta el año 2005, cuando Le Meur et al.[14] propusieron el primer modelo para determinar las regiones más llamativas en secuencias de vídeo. Como veremos en esta sección, su sencilla idea aún sigue siendo usada en los mejores algoritmos actuales.

En su artículo, Le Meur et al.[14] se dieron cuenta, tras analizar numerosos experimentos con *eye-trackers*, de que la saliencia final de una secuencia de frames venía determinada principalmente por dos factores: estáticos (de los que hemos hablado en la sección 1.1) y dinámicos (la percepción del movimiento).

Por aquel entonces las mejores técnicas para estimar el movimiento de los objetos de un vídeo consistían en una búsqueda jerárquica de *block-matching*. Es decir, usando una estructura piramidal gaussiana, el algoritmo trataba de encontrar el bloque en el siguiente frame que más se pareciese a cada bloque del frame actual.

Además, idearon un modelo de fusión de los mapas de saliencia espacial y temporal muy interesante, que sirvió de base para futuras mejoras de otros investigadores.

Como podemos observar en la ecuación 1.1 y la figura 1.5, la fusión da toda la importancia a la saliencia estática cuando apenas hay movimiento en la escena; progresivamente más importancia a la saliencia dinámica según el movimiento total incrementa; y nuevamente sólo utiliza la saliencia espacial cuando la escena contiene excesivo movimiento.

$$S(p) = \alpha S^T(p) + (1-\alpha)S^E(p) + \beta S^T(p)S^E(p) \tag{1.1}$$

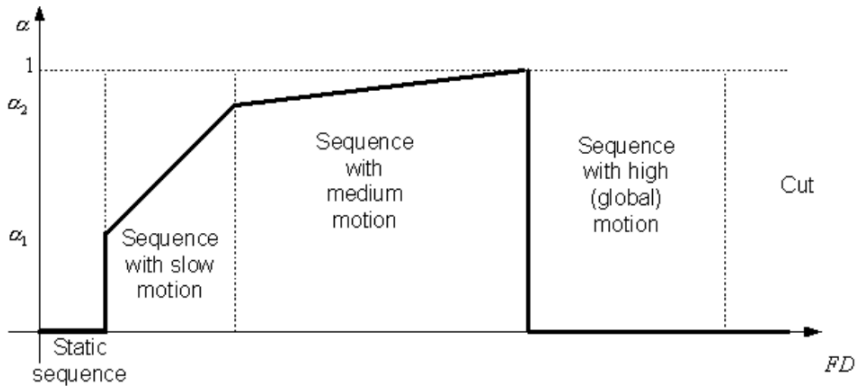


Figura 1.5. Parámetros para la fusión de los mapas de saliencia espacial y temporal propuestos por Le Meur et al.[14].

Nótese que p representa las coordenadas de un píxel; $S^T(p)$ el valor de la saliencia temporal -es decir, el movimiento estimado- en el píxel p; y $S^E(p)$ la saliencia espacial obtenida por alguno de los métodos descritos en la sección 1.1. En la figura 1.5, *FD* representa la diferencia total entre dos frames consecutivos.

Poco después de la publicación del artículo de Le Meur et al.[14], varios investigadores aportaron nuevos modelos. Zhai y Shah[15] propusieron mejorar la estimación de movimiento usando un KLT tracker[16]. Sin embargo, el método de Kanade-Lucas-Tomasi sólo identifica el movimiento de puntos particulares del frame, por lo que no se consigue una precisión a nivel de píxel, sino de *caja delimitadora*.

Por su parte, Marat et al.[17] plantearon un nuevo enfoque: en lugar de realizar aproximaciones matemáticas, intentaron proponer un modelo que simulase el funcionamiento del cerebro desde un punto de vista neurobiológico. Al final, su modelo también recopilaba dos mapas de saliencia -espacial y temporal- y se enfrentaba al difícil problema de cómo fusionarlos.

En su publicación de 2008[17], concluyeron que de entre los distintos algoritmos de fusión que probaron, el AND lógico ($S(p) = S^T(p) \times S^E(p)$) era el que más se aproximaba a los datos recogidos por el *eye-tracker*. Al año siguiente, la continuación de sus investigaciones[18] desembocó en un algoritmo de fusión más elaborado, similar a la ecuación 1.1, escogiendo el valor de $1 - \alpha = \text{máx}(S^E)$.

Durante los siguientes años, los esfuerzos se centraron en mejorar la forma en que los dos mapas de saliencia se combinaban en un único mapa final. En esta línea fueron los trabajos de Liu et al.[19] y Chamaret et al.[20], que a pesar de incrementar la eficacia de los algoritmos de detección de saliencia en vídeos, seguían teniendo mucho margen de mejora.

Motivados por el algoritmo de saliencia espacial de Hou y Zhang[8] mencionado en la sección 1.1, Guo et al.[21] plantearon que la saliencia no viene determinada por la amplitud del espectro de la Transformada de Fourier de una imagen, sino más bien por su fase. Para calcular la saliencia espacio-temporal, su método extiende la Transformada de Fourier en 2D a la TF de Cuaterniones, donde el valor de cada píxel viene represantado por un cuaternión compuesto de intesidad, color y velocidad. Gracias a la posibilidad de calcular la TF de manera muy eficiente, este algoritmo reduce enormemente el tiempo de cálculo de la saliencia de un vídeo.

Por último, el modelo más eficaz desarrollado hasta ahora es el de Zhong et al.[22]. A pesar de utilizar un algoritmo de fusión tan simple como la función MAX ($S(p) = \text{máx}[S^T(p), S^E(p)]$), debe su éxito a la mejora que proponen sobre el cálculo clásico del flujo óptico para estimar el movimiento.

Además de mejorar la eficiencia de la implementación reutilizando las pirámides gaussianas de la saliencia estática para determinar el movimiento de la imagen, obtienen mejores resultados al calcular el flujo óptico entre un frame (i) y el anterior del anterior (i-2). En la figura 1.6 hemos incluido el modelo de saliencia espacio-temporal propuesto por Zhong et al.[22].

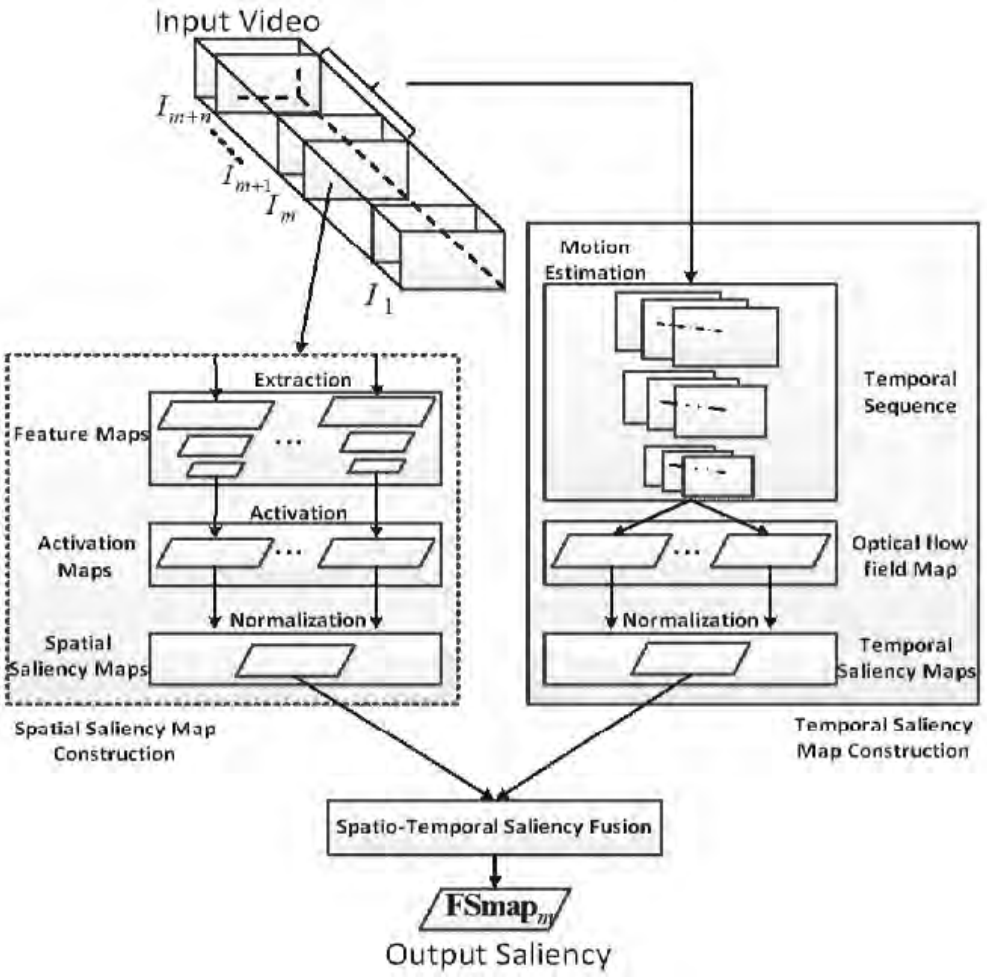


Figura 1.6. Algoritmo para el cálculo de la saliencia espacio-temporal planteado por Zhong et al.[22].

Para concluir la sección, hemos querido mencionar dos aplicaciones actuales de la detección de saliencia en secuencias de vídeo. Por un lado, Rapantzikos et al.[23] introdujeron el concepto de *voxel* (similar al de segmento en una imagen pero aplicado a un conjunto de frames), gracias al cual pudieron elaborar un algoritmo de clasificación de vídeos basado en el análisis de las características de sus *voxels*. En su artículo, utilizaron como ejemplo vídeos deportivos para demostrar la validez de su clasificador. En la figura 1.7 podemos ver la diferencia entre los volúmenes existentes en un vídeo de natación frente a uno de tenis.

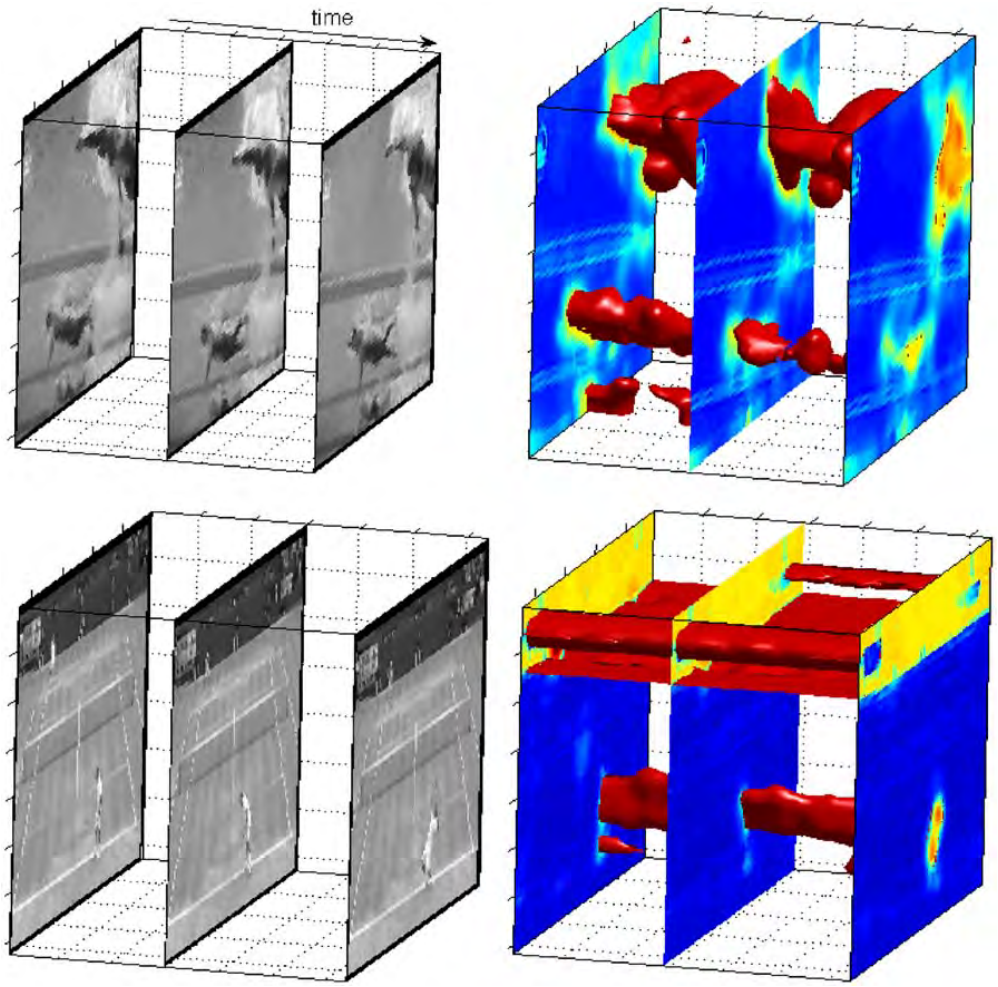


Figura 1.7. Aplicación de la saliencia espacio-temporal para clasificar vídeos[23].
Los volúmenes en rojo representan *voxels* con valores de saliencia muy elevados.

Como otro ejemplo del uso de la saliencia dinámica en la actualidad, hemos incluido la figura 1.8. En ella se muestra el resultado del algoritmo de *video retargeting* (reducción de la resolución de un vídeo manteniendo sus regiones relevantes siempre visibles) desarrollado por Lu et al.[24]. La principal aplicación de este algoritmo podría ser para generar vistas previas de imágenes de gran tamaño o para adaptar vídeos en formato 16:9 a pantallas 4:3.

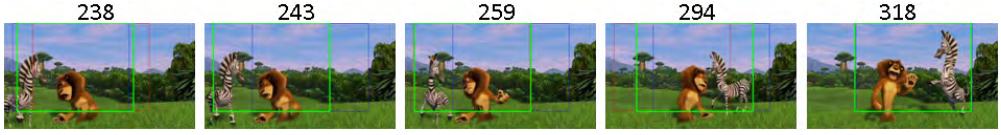


Figura 1.8. Uso de la saliencia espacio-temporal para *video retargeting*[24].
El algoritmo intenta mantener encuadrado (en verde) lo más relevante del vídeo original.

1.3. Estimación de movimiento

Durante varias décadas la estimación de los vectores movimiento ha sido un tema de gran interés para la comunidad investigadora en visión artificial. Las aplicaciones de estas técnicas en la robótica y la codificación de vídeo son enormes. En esta sección vamos a mencionar los principales métodos que permiten estimar el movimiento, para posteriormente centrarnos en la técnica que hasta ahora está dando los mejores resultados en los algoritmos de saliencia dinámica: el flujo óptico.

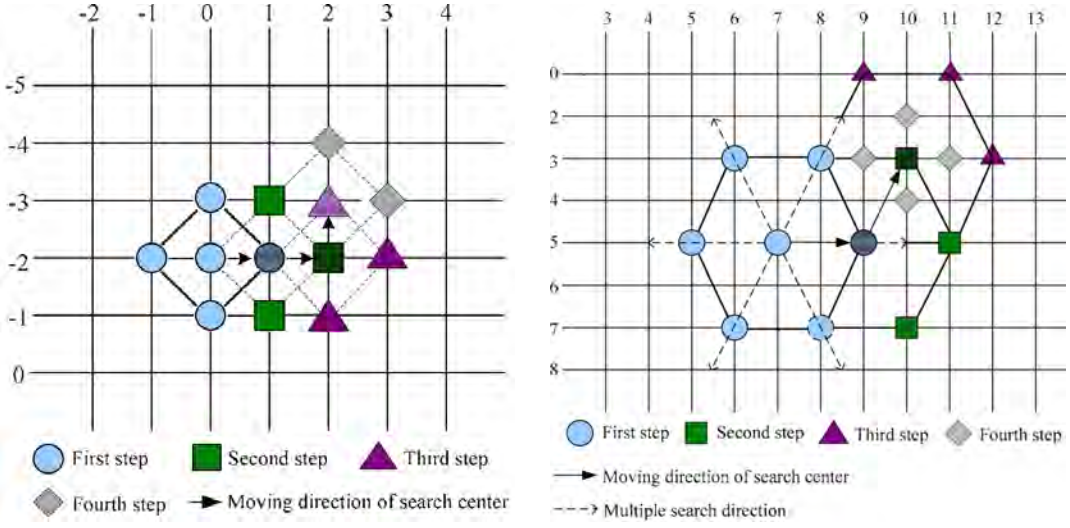
Existen fundamentalmente dos maneras de calcular los vectores de movimiento: trabajando a nivel de píxel(es) -métodos directos- o a nivel de características -indirectos-. A este último tipo pertenece el modelo de Nistér[25], que se basa en la obtención de características, como la detección de esquinas, seguido de una búsqueda de correspondencias entre características de frames consecutivos. Para este último paso se emplea alguna función estadística -que suele incluir RANSAC[26,27]- que se encarga de eliminar falsas correspondencias que no coincidan con el movimiento real. El algoritmo es bastante más rápido que los métodos directos pero a cambio tiene peor precisión, en parte por no trabajar a nivel de píxel.

Dentro de los métodos directos de estimación de movimiento debemos destacar el *block-matching* y el flujo óptico. Se han dedicado muchos esfuerzos a perfeccionar ambas técnicas, pero el *block-matching* se ha encaminado más hacia temas de codificación de vídeo, en los que a menudo se buscan bloques similares en un radio de varios fotogramas. Gracias a eso, los algoritmos actuales son muy eficientes para la compresión de vídeo, sin embargo no rinden tan bien cuando se trata de calcular la velocidad de cada bloque en un frame respecto del siguiente.

La idea detrás del *block-matching* es simple. No obstante, diseñar algoritmos eficientes requiere superar un gran problema: la complexidad. Es por esto que la mayoría de los esfuerzos de la investigación en *block-matching* han ido dirigidos a reducir el número de veces que cada bloque debe ser comparado con otros hasta encontrar la mejor coincidencia.

A pesar de centrarnos en el flujo óptico en esta sección, hemos querido mencionar las sucesivas mejoras más importantes en los algoritmos de búsqueda para *block-matching*, como son: la búsqueda en cruz[28], jerárquica[29], en 3 pasos[30,31], en 4 pasos[32], en diamante[33,34] (figura 1.9a) y en hexágono[35] (figura 1.9b).

Una vez comentado el estado del arte en *block-matching*, vamos a repasar la evolución de los modelos de estimación de movimiento basados en el cálculo del flujo óptico.



(a) Patrón de búsqueda en diamante[33,34] **(b)** Patrón de búsqueda en hexágono[35]

Figura 1.9. Algoritmos de búsqueda de *block-matching* con grandes resultados.

El primer algoritmo formal para determinar la velocidad en la imagen a partir del flujo óptico fue propuesto por Horn et al.[36] en 1981. Según su modelo, para un intervalo Δt suficientemente pequeño, la intensidad de un mismo punto en ambos fotogramas debe permanecer constante. Esto se puede describir matemáticamente como indica la ecuación 1.2, llamando Δx y Δy al desplazamiento de dicho punto de un frame a otro:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{1.2}$$

Aplicando una aproximación local de la serie de Taylor a la señal de la imagen, obtenemos la ecuación 1.3, en la que T.O.S. significa términos de orden superior:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + T.O.S. \tag{1.3}$$

Combinando las ecuaciones 1.2 y 1.3, llegamos a:

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{1.4}$$

O lo que es lo mismo:

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0 \tag{1.5}$$

Denominando v_x y v_y a las componentes de la velocidad -flujo óptico de $I(x, y, t)$-; I_x, I_y e I_t a las derivadas de la intensidad con respecto a x, y y t:

$$I_x v_x + I_y v_y = -I_t \tag{1.6}$$

Equivalentemente, a menudo la ecuación 1.6 se reescribe como:

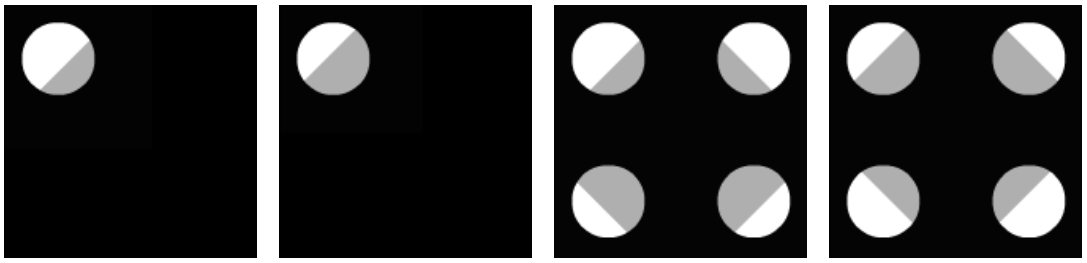
$$\nabla I^T \cdot \vec{V} = -I_t \tag{1.7}$$

Como podemos ver, tenemos una ecuación y dos incógnitas, por lo que es necesario añadir alguna restricción más aparte de la constancia en intensidad. De hecho, la elección de estas ecuaciones adicionales ha sido el núcleo de la investigación en flujo óptico desde sus orígenes. En su modelo, Horn et al.[36] plantean una segunda restricción de suavizado, ya que salvo en puntos singulares (como cuando haya oclusión) la velocidad de un punto será similar a la de sus vecinos. Por ello tratan de minimizar la magnitud al cuadrado del gradiente de la velocidad:

$$\left(\frac{\partial v_x}{\partial x}\right)^2 + \left(\frac{\partial v_x}{\partial y}\right)^2 \qquad y \qquad \left(\frac{\partial v_y}{\partial x}\right)^2 + \left(\frac{\partial v_y}{\partial y}\right)^2 \tag{1.8}$$

Además, debido a que una imagen muestra una proyección en dos dimensiones de objetos en 3D, múltiples patrones pueden producir la misma proyección, lo que se conoce como el "problema de apertura". McDermott et al.[37] describen muy bien en qué consiste este problema, proporcionando numerosos ejemplos y proponiendo métodos para discriminar *outliers*. Por ejemplo, la figura 1.11 muestra tres movimientos distintos que se perciben como el mismo debido a la oclusión.

En la figura 1.10 podemos ver otro ejemplo de este problema y cómo puede ser solucionado en casos simples. Si nuestro enfoque es local -es decir, sólo observamos lo que ocurre en una pequeña región- podemos sacar conclusiones erróneas del movimiento. Así, en las imágenes a y b puede parecer que tenemos una esfera rotando hacia el noroeste. Sin embargo, si ampliamos nuestro enfoque y miramos globalmente al fotograma (c y d) podemos utilizar la información de otros puntos para deducir el movimiento correcto -en este caso, un rombo que sube hacia arriba.



(a) Frame 1 Enfoque local | **(b)** Frame 2 Enfoque local | **(c)** Frame 1 Enfoque global | **(d)** Frame 2 Enfoque global

Figura 1.10. Ejemplo práctico del problema de apertura.
Para determinar el movimiento real bajo la existencia de contornos iso-brillantes es necesario observar la imagen en su conjunto, desde un punto de vista más global.

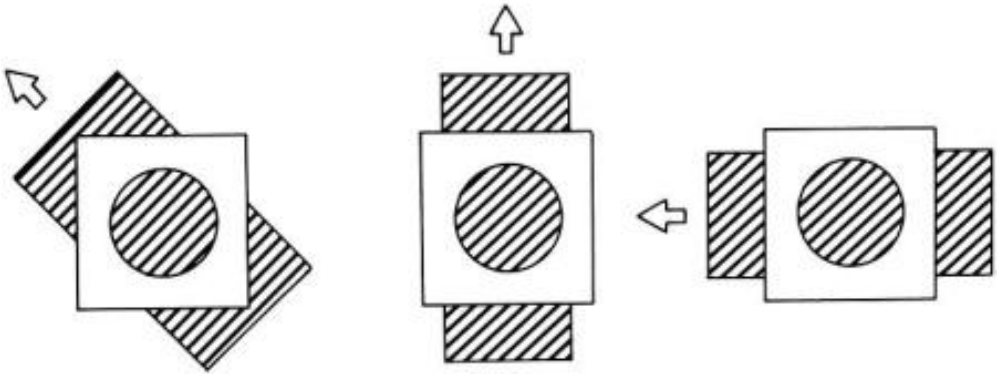


Figura 1.11. Otro ejemplo práctico del problema de apertura.

Durante más de una década, varios investigadores intentaron mejorar la exactitud de la estimación del movimiento proponiendo diversas soluciones al problema de apertura: utilizando filtros espacio-temporales (Heeger[38]), derivando propiedades teóricas del flujo óptico (Verri y Poggio[39]) o calculando y analizando sus distribuciones de probabilidad (Simoncelli et al.[40]).

En 1994, Proesmans et al.[41] plantearon una mejora significativa. Primero, detectaban grandes discontinuidades en el flujo óptico. Después, esa información era enviada de vuelta a la restricción de suavizado mediante una retroalimentación no linear, que suavizaba el flujo manteniendo intactas las discontinuidades.

No obstante, por aquel entonces era difícil determinar si un algoritmo realmente superaba a sus predecesores por la ausencia de *ground truth*. En ese sentido, Barron et al.[42] y Galvin et al.[43] hicieron un gran trabajo al generar varios vídeos sintéticos y analizar la eficacia de los principales modelos propuestos hasta el momento.

Gracias a la tecnología actual, ahora disponemos de varias bases de datos con *ground truth* correspondiente a vídeos reales. Por un lado, Baker et al.[44] utilizaron una sustancia fluorescente en el espectro ultravioleta para grabar una serie de vídeos de los que se puede conocer con exactitud su flujo óptico. Además, propusieron el código de color de la figura 1.12a para visualizar el movimiento sin usar vectores. Por otro lado, Liu et al.[45] diseñaron una aplicación que permite a los usuarios anotar el *ground truth* de cualquier vídeo, como se puede ver en la figura 1.12 (b, c y d).

Recientemente, se han propuesto varios modelos que utilizan pirámides gaussianas junto con las técnicas de conservación de discontinuidades para, yendo de un nivel con poco detalle hacia niveles más detallados, solucionar el problema de apertura de una manera muy eficaz. En este aspecto cabe mencionar los algoritmos de Brox et al.[46], Bruhn et al.[47], Fleet y Weiss[48] y Alvarez et al.[49].

(a) Código de color[44] **(b)** Frame de un vídeo **(c)** Segmentación **(d)** *Ground truth*

Figura 1.12. Proceso de generación de *ground truth* para flujo óptico[45]. Para cada frame (b), el usuario asiste al programa en la segmentación (c) y así se genera automáticamente el *ground truth* (d).

Por último, consideramos importante destacar la precisión lograda por Sun et al.[50]. Gracias a una técnica que mejora el filtrado de mediana de los algoritmos anteriores, actualmente se sitúa como el mejor modelo de estimación de movimiento según la base de datos de Baker et al.[44]. De cara al futuro, en su artículo sugieren usar información de más de dos frames al mismo tiempo para mejorar las estimaciones del flujo óptico.

1.4. Segmentación de imágenes

En esta sección vamos a hacer un breve repaso de las distintas técnicas de actualidad que se utilizan para segmentar imágenes. Hemos necesitado incluir un paso de segmentación en nuestro modelo, como ya explicaremos más adelante, por dos motivos. En primer lugar, acabamos de ver que los algoritmos de estimación de movimiento aún no son perfectos y tienen problemas en las discontinuidades. Dividiendo la imagen en segmentos y asignando a cada uno su velocidad media, logramos reducir esos errores. Pero además, el *ground truth* del que disponemos proviene de un *eye tracker*, que únicamente nos indica a qué píxel miraba el usuario. Para obtener información más relevante, hemos decidido segmentar la imagen y atribuir la saliencia a los segmentos cercanos al centro de la mirada.

En realidad la segmentación puede verse como un caso de *clustering*. El algoritmo más popular para este tipo de problemas es el conocido k-means[51]. Tras una inicialización aleatoria, dos pasos se repiten iterativamente hasta que los cambios de un paso al siguiente se encuentren por debajo de un umbral fijado. En el primer paso, se asigna cada punto al *cluster* cuyo centro esté más cercano. Después, se recalcula el centro de cada *cluster* en función de los nuevos puntos. Una de las primeras implementaciones en Fortran fue la de Hartigan y Wong[52].

K-means ha servido de base para muchos algoritmos posteriores, pero cuenta

con varios problemas: el número de *clusters* debe ser conocido con anterioridad. Además, se pueden obtener distintos resultados en función de la inicialización, lo que dificulta la repetibilidad de los experimentos. Por último, el algoritmo tiene asegurada la convergencia, pero en un mínimo local, no global. Esto suele solucionarse ejecutando el algoritmo varias veces, pero no siempre es suficiente.

Durante los últimos años, se han explorado otras vías para la segmentación de imágenes. El sistema que nosotros hemos usado, abreviado como EDISON[53], utiliza filtros de mean shift (Comaniciu y Meer[54]) y detección de bordes (Meer y Georgescu[55]) para producir segmentaciones como la de la figura 1.13 (a y b).

Al no existir *ground truth* sobre segmentación -debido a la subjetividad de la tarea y el nivel de detalle deseado-, es difícil determinar qué algoritmos son mejores que otros. Como discutiremos más adelante, nos hemos decantado por este sistema por su sencilla implementación, su velocidad y su ligera sobre-segmentación, que nos permite tener el nivel de detalle deseado para la estimación de saliencia.

Por último, queríamos mencionar el algoritmo de Grundmann et al.[56], que permite segmentar secuencias de vídeo obteniendo información de múltiples frames a la hora de detectar regiones similares. Sin embargo, su tiempo de ejecución supone aún un gran impedimento -unos 40 segundos a resoluciones de 640x480 frente a unos 3s con EDISON. La figura 1.13 (c y d) muestra un ejemplo.

(a) Imagen original **(b)** Segmentada **(c)** Frame original **(d)** Segmentado

Figura 1.13. Ejemplos de segmentación de imagen[53] y vídeo[56].

1.5. Aprendizaje máquina

Para este proyecto, hemos querido utilizar alguna técnica de aprendizaje máquina como complemento adicional a nuestro algoritmo. Es decir, este último paso que vamos a explicar a continuación no forma parte del modelo de detección de saliencia espacio-temporal.

Nuestro algoritmo únicamente se encarga de obtener los mapas de características relacionados con la saliencia espacial y el movimiento. Para combinar la información de estos mapas y dar lugar a un único mapa final, hemos utilizado

técnicas de aprendizaje máquina que estimen la ponderación que debe recibir cada uno para que la salida sea lo más parecida posible a nuestro *ground truth*.

A pesar de que conocemos entornos muy potentes que ofrecen todo tipo de técnicas de aprendizaje máquina (como Weka), puesto que todo el proyecto se ha desarrollado en Matlab, hemos decidido ceñirnos a las alternativas que ofrece este programa a través de su *toolbox* de Redes Neuronales para facilitar su integración con el resto del sistema.

Gracias a su sencilla interfaz gráfica, hemos podido diseñar una red neuronal de tipo MLP[57] (Multi-Layer Perceptron), que suelen generalizar muy bien la información aprendida y están extensamente documentadas. La figura 1.14 muestra un ejemplo de un MLP con dos entradas y 5 neuronas en su capa *oculta*.

Como veremos en la sección 3.2, nuestro MLP constará de tres entradas (una por característica) y una salida, que tendrá valores cercanos a 0 cuando un segmento no sea llamativo y próximos a 1 cuanto más saliente sea. Además, experimentaremos con diferentes tamaños de capa oculta para observar los resultados.

Por último, queremos hacer referencia al algoritmo de entrenamiento utilizado. En su *toolbox*, Matlab incluye implementaciones de los algoritmos más populares: *Levenberg-Marquardt*[58,59], *Bayesian regulation*[60,61], *Resilient backpropagation*[62] y *Scaled conjugate gradient*[63].

Después de bastantes pruebas e intentos fallidos, tuvimos que descartar los dos primeros métodos de entrenamiento debido a que la enorme cantidad de memoria requerida acababa colgando nuestro ordenador antes de que el proceso finalizase. Asimismo, de entre los dos algoritmos restantes, seleccionamos el *Scaled conjugate gradient*[63] debido principalmente a su mayor rapidez y tasa de acierto en casi la totalidad de nuestros experimentos.

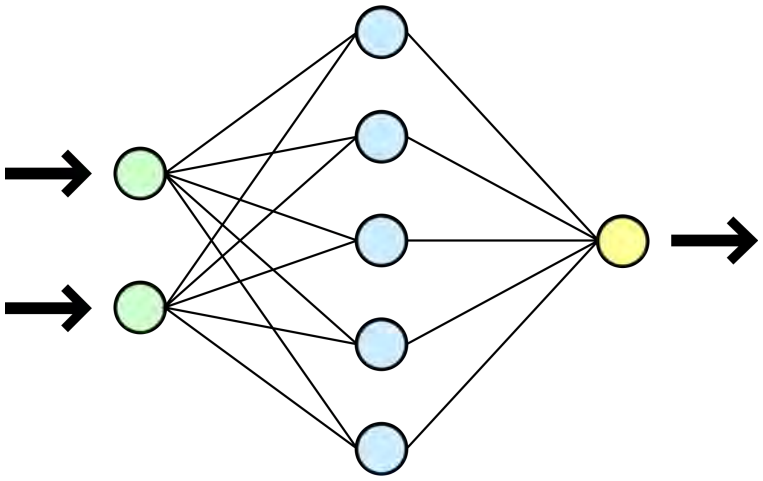


Figura 1.14. Ejemplo de un MLP con 5 neuronas en su capa oculta.

Capítulo 2
Descripción del modelo

En este capítulo vamos a explicar los pasos que hemos seguido para construir nuestro modelo de saliencia espacio-temporal.

En primer lugar, analizaremos las diferentes alternativas de las que disponíamos para calcular la saliencia estática. Discutiremos las ventajas e inconvenientes de los principales modelos y justificaremos nuestra elección, además de enseñar ejemplos de los resultados del algoritmo escogido.

Posteriormente, comentaremos la implementación del cálculo del flujo óptico que hemos empleado, explicando las magnitudes extraídas de la estimación de movimiento y mostrando los resultados que produce en secuencias de ejemplo.

Por último, hablaremos de la herramienta de segmentación que hemos utilizado y el modelo que hemos ideado para combinar la información de estas tres fuentes.

2.1. Cálculo de la saliencia espacial

2.1.1. Elección de la implementación

En la sección 1.1 hemos analizado el estado del arte en algoritmos de saliencia estática. Para nuestro proyecto queríamos, por supuesto, un algoritmo cuyos resultados estuviesen entre los mejores. Pero además, hemos empleado criterios adicionales como el nivel de configuración de parámetros que ofrece cada uno o la compatibilidad multiplataforma.

En este sentido, debido a las dificultades que a veces encontramos al compilar código `C++` en distintos sistemas operativos, la extensa documentación de Matlab y sus potentes librerías ya desarrolladas, hemos preferido implementaciones en este último lenguaje a pesar de ser claramente más lentas.

De hecho, queremos dejar claro que nuestro objetivo es conseguir un modelo de saliencia espacio-temporal que supere a los mejores algoritmos de saliencia estática aplicados frame a frame. Como se menciona en la sección 4.3, la optimización del código para reducir el tiempo de ejecución queda abierto para el futuro.

Tras identificar los mejores algoritmos actuales en este campo (sección 1.1), hemos rellenado el cuadro 2.1 con los datos de las implementaciones oficiales de cada modelo de saliencia estática.

Autor(es)	**Leng.**	**Link**
Cheng et al.[12]	`C++`	`mmcheng.net/effisalobj/`
Perazzi et al.[11]	`C++`	`graphics.ethz.ch/~perazzif/saliency_filters/`
Judd et al.[9]	Matlab	`people.csail.mit.edu/tjudd/WherePeopleLook/`
Hou y Zhang[8]	Matlab	`www.klab.caltech.edu/~xhou/projects/spectralResidual/spectralresidual.html`
Walther y Koch[64]	Matlab	`www.saliencytoolbox.net/`
Itti y Koch[5]	`C++`	`ilab.usc.edu/toolkit/`

Cuadro 2.1. Datos sobre varias implementaciones de modelos de saliencia estática.

A continuación vamos a comentar las distintas alternativas, así como justificar la elección del `SaliencyToolBox` de Walther y Koch[64], basado en el modelo original de Itti et al.[4].

En primer lugar, las implementaciones de Cheng et al.[12] y Perazzi et al.[11] están en `C++`. A pesar de conseguir los mejores resultados hasta el momento, la ausencia de librerías que nos permitiesen integrar este algoritmo con el resto del modelo iba a suponer un esfuerzo adicional que nos alejaría de nuestro objetivo fundamental.

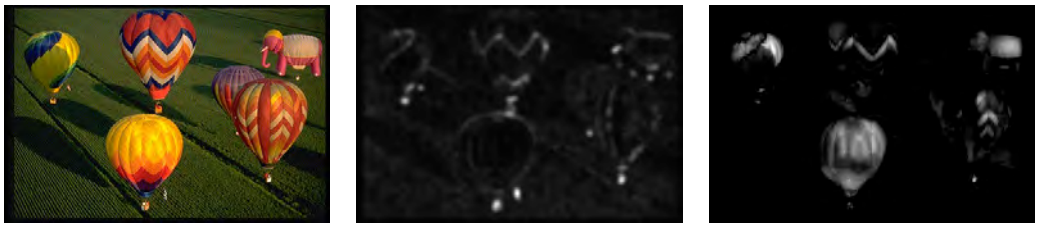
Por su parte, el algoritmo de Judd et al.[9] está muy centrado en la detección de personas y rostros. Como comentamos en la sección 1.1, este modelo fue entrenado con una base de datos en la que la mayoría de imágenes contenían personas. Por este motivo, nos hemos decantado por un algoritmo más genérico, válido tanto para secuencias con seres humanos como con paisajes.

El algoritmo de Hou y Zhang[8] podría ser una buena opción. Implementado con seis líneas en Matlab, es muy rápido y sencillo de entender. Sin embargo, el mapa en escala de grises que produce no es tan bueno como el de otros algoritmos (véase la figura 2.1b).

Por último, nos quedan por evaluar el `SaliencyToolBox` de Walther y Koch[64] y el `iNVT` (iLab Neuromorphic Vision `C++` Toolkit) de Itti y Koch[5]. Como explica

Walther[65] en el apéndice B de su tesis doctoral -llamado `SaliencyToolBox`-, `iNVT` es una herramienta muy potente que contiene una funcionalidad que va mucho más allá del cálculo de la saliencia espacial. Con aproximadamente 360 000 líneas de código, se hace muy difícil experimentar con la herramienta para alguien que no la ha programado.

En cambio, el `SaliencyToolBox`, implementado completamente en Matlab y actualizado por última vez en julio de 2013 (versión 2.3), permite familiarizarse rápidamente con el software a través de una sencilla interfaz gráfica desde la que se pueden ajustar todos los parámetros (figura 2.2).



(a) Imagen original **(b)** Hou y Zhang[8] **(c)** `SaliencyToolBox`[64]

Figura 2.1. Comparación de las implementaciones candidatas a ser utilizadas.

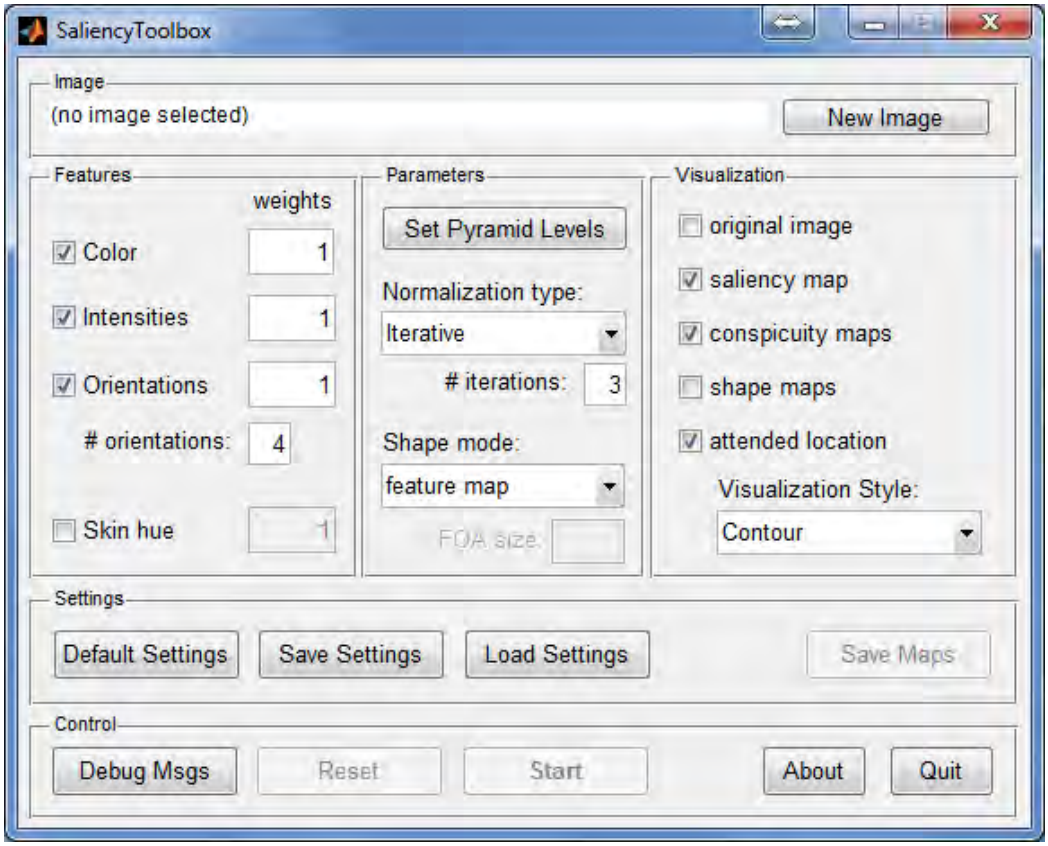


Figura 2.2. Interfaz gráfica del `SaliencyToolBox` en Matlab.

A la vista de los resultados de los algoritmos de Hou y Zhang[8] y Walther y Koch[64], como los mostrados en la figura 2.1, y por todos los motivos mencionados anteriormente, nos hemos decantado por el `SaliencyToolBox` para obtener el mapa de saliencia espacial.

2.1.2. Descripción del modelo

Una vez elegido el modelo de saliencia estática que vamos a emplear, hemos querido dedicar unos párrafos a recordar su funcionamiento. Para ello nos vamos a ayudar de la figura 1.1, que explica a grandes rasgos el algoritmo de Itti et al.[4] sobre el que se basa el `SaliencyToolBox`.

De los cuatro pasos descritos en la sección 1.1, vamos a centrarnos en los tres primeros, ya que la parte final que determina la región más llamativa a partir del mapa de saliencia será reemplazada por un paso que combine dicho mapa con la información obtenida del flujo óptico. Vayamos al grano.

Cuando el algoritmo recibe una imagen, lo primero que hace es crear lo que se conoce como una pirámide gaussiana. Con la posibilidad de elegir el número de niveles deseados, cada nivel guarda una copia de la imagen del nivel anterior, pero reduciendo sus dimensiones a la mitad. Por ejemplo, si tuviésemos una imagen de 400x300 píxeles, el primer nivel almacenaría la imagen original, el segundo la reescalaría a 200x150, el tercero a 100x75, y así sucesivamente.

Una vez creada esta pirámide, el algoritmo realiza una serie de filtrados para obtener los siguientes datos en cada nivel: variación de intensidad, de color (rojo-verde y azul-amarillo) y de orientación -que parecen ser los estímulos principales que activan nuestra atención-.

Después, se crea un mapa de características en cada escalón de la pirámide usando la técnica conocida como *center-surround*: en un nivel c se restan la variación en dicho nivel y la de varios niveles por encima, $c + \delta$ (con esto se consiguen valores altos cuando los objetos destacan respecto de su alrededor). Típicamente, $c \in \{1, 2, 3, 4\}$ y $\delta \in \{3, 4\}$.

Esto va seguido de un paso de normalización, que pretende dar más importancia a los mapas en los que el valor máximo destaque mucho y menos a los que contengan numerosos picos muy similares. Posteriormente, los mapas de todos los niveles se reescalan al nivel más bajo de la pirámide y se suman, dando lugar a un único mapa por característica.

Por último, los tres mapas se combinan linealmente -con una ponderación por defecto de 33 % por característica- para producir el mapa final de saliencia.

2.1.3. Configuración de los parámetros

Como hemos podido ver en la figura 2.2, el `SaliencyToolBox` nos proporciona un alto nivel de configuración. Algunos parámetros -los de la pestaña 'Visualización'- sólo afectan a la interfaz gráfica. Otros, como el 'modo de forma' -que indica qué mapa se utilizará para obtener la región saliente una vez determinado el píxel más llamativo-, afectan a la segmentación posterior al cálculo de la saliencia, y puesto que nosotros solo vamos a usar el mapa en escala de grises no nos afectarán.

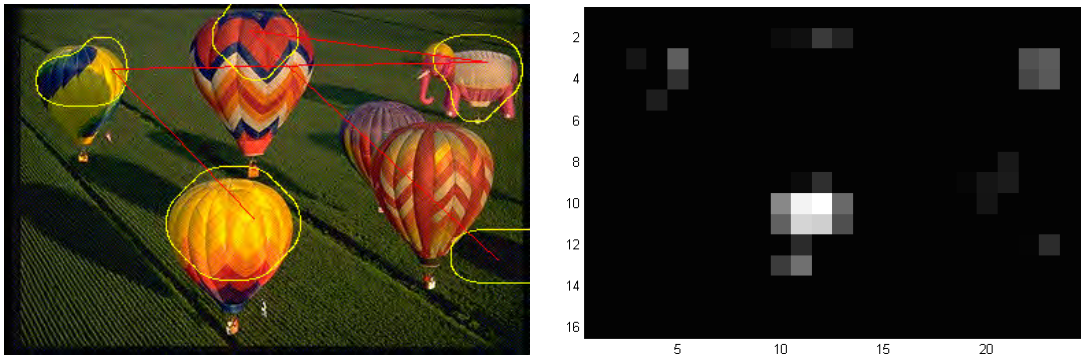
Así pues, los parámetros que nos quedan por configurar son los niveles de la pirámide gaussiana y el tipo de normalización que aplicar a los mapas de características antes de combinarlos para dar lugar al mapa final de saliencia. La figura 2.4 muestra las 5 regiones más salientes de una imagen de ejemplo con todos los parámetros por defecto. A continuación veremos si podemos mejorar los resultados configurando la herramienta para que se ajuste a nuestras necesidades.

En la sección 2.2 veremos que el flujo óptico nos proporciona un vector de movimiento para cada píxel. Por eso, hemos decidido modificar el parámetro de las pirámides gaussianas que indica a qué nivel de la pirámide calcular el mapa final de saliencia. Por defecto, este valor es 5, lo cual significa que el mapa de saliencia tendrá unas dimensiones 2^4 veces más pequeñas que la imagen original. Tras cambiar este parámetro a 1 -así el mapa final tendrá la misma resolución que la imagen de entrada- los resultados obtenidos para la fotografía de los globos se enseñan en la figura 2.5.

Por otro lado, la figura 2.3 muestra el mapa de saliencia resultante en función del número de iteraciones de normalización. Por defecto, este dato valía 3 (d). En las imágenes a, b y c se puede ver cómo varía la salida al modificar este parámetro. Después de hacer varias pruebas con diversas imágenes, hemos observado que los resultados tanto sin normalizar como con 3 iteraciones no eran buenos (la salida tenía o mucha energía -casi todo era saliente- o muy poca -casi nada destacaba-). Al final nos decidimos por realizar solo una iteración porque así el mapa final tiene algo más de detalle que con dos pasos de normalización (véase figura 2.7).

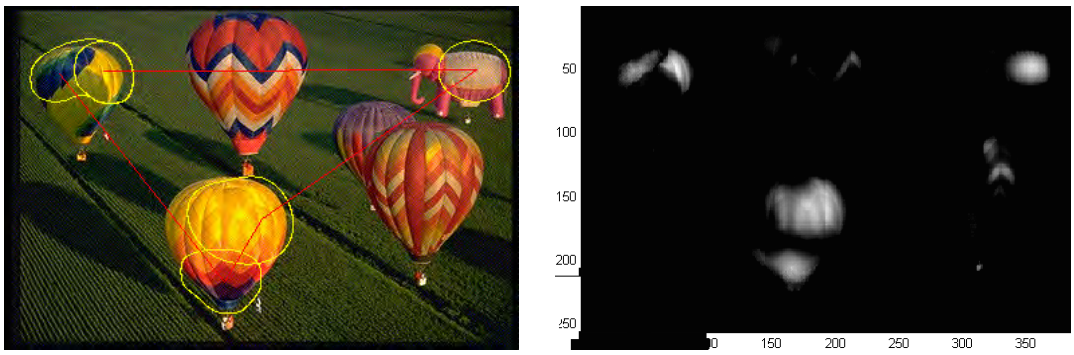
(a) Sin normalizar **(b)** 1 iteración **(c)** 2 iteraciones **(d)** 3 iteraciones

Figura 2.3. Resultados con diferente número de iteraciones de normalización.



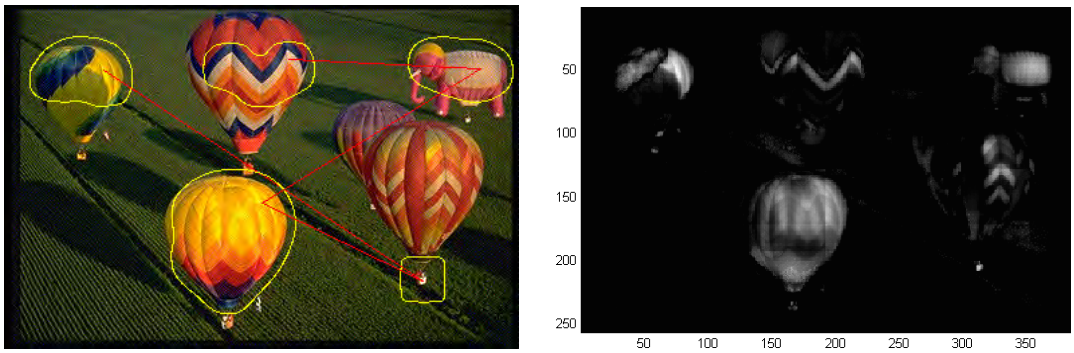
(a) 5 regiones más salientes **(b)** Mapa de saliencia

Figura 2.4. Resultados del `SaliencyToolBox` con los parámetros por defecto. Por defecto, el mapa de saliencia se calcula a una resolución 16 veces inferior a la original y con 3 iteraciones de normalización.



(a) 5 regiones más salientes **(b)** Mapa de saliencia

Figura 2.5. Resultados del `SaliencyToolBox` tras ajustar la resolución de salida. Los resultados mejoran si configuramos el `SaliencyToolBox` para obtener el mapa de saliencia a la resolución de la imagen de entrada.



(a) 5 regiones más salientes **(b)** Mapa de saliencia

Figura 2.6. Resultados del `SaliencyToolBox` con nuestra configuración final. Los mejores resultados se obtienen si además pasamos de 3 iteraciones a un único paso de normalización.

(a) Imagen original **(b)** 1 iteración **(c)** 2 iteraciones

Figura 2.7. Resultados con una y dos iteraciones de normalización.
Preferimos una sola iteración (b) porque se obtiene algo más de detalle que con dos (c).

2.1.4. Resultados y ejemplos

Para concluir la sección sobre el modelo de saliencia espacial, hemos querido añadir la siguiente figura, que muestra los mapas de saliencia obtenidos para varias imágenes de prueba.

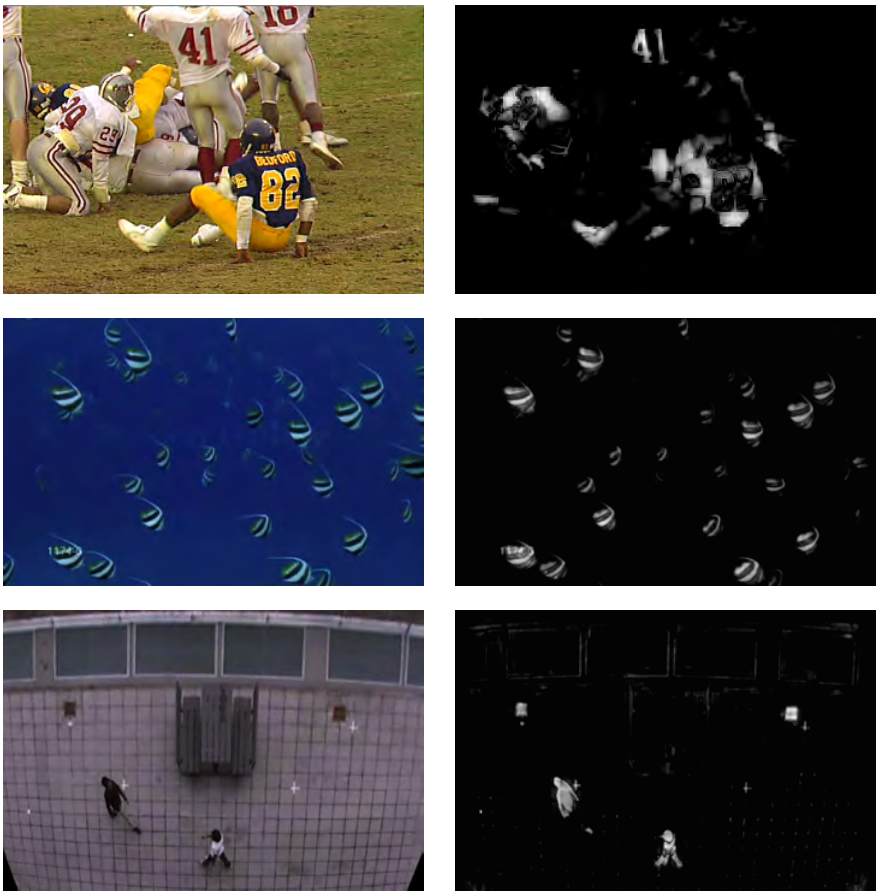


Figura 2.8. Resultados del `SaliencyToolBox` ante varias imágenes de prueba.

2.2. Cálculo del flujo óptico

2.2.1. Elección de la implementación

En la misma línea que la sección anterior, para el cálculo del flujo óptico hemos buscado implementaciones en Matlab que ofrezcan los mejores resultados posibles.

De los algoritmos mencionados en el estado del arte en estimación del movimiento (sec. 1.3), sólo hemos encontrado cuatro en este lenguaje. Dos de ellos, el de Horn/Schunck[36] y el de Lucas/Kanade[47], están ya implementados en Matlab como parte del *Computer Vision System Toolbox*.

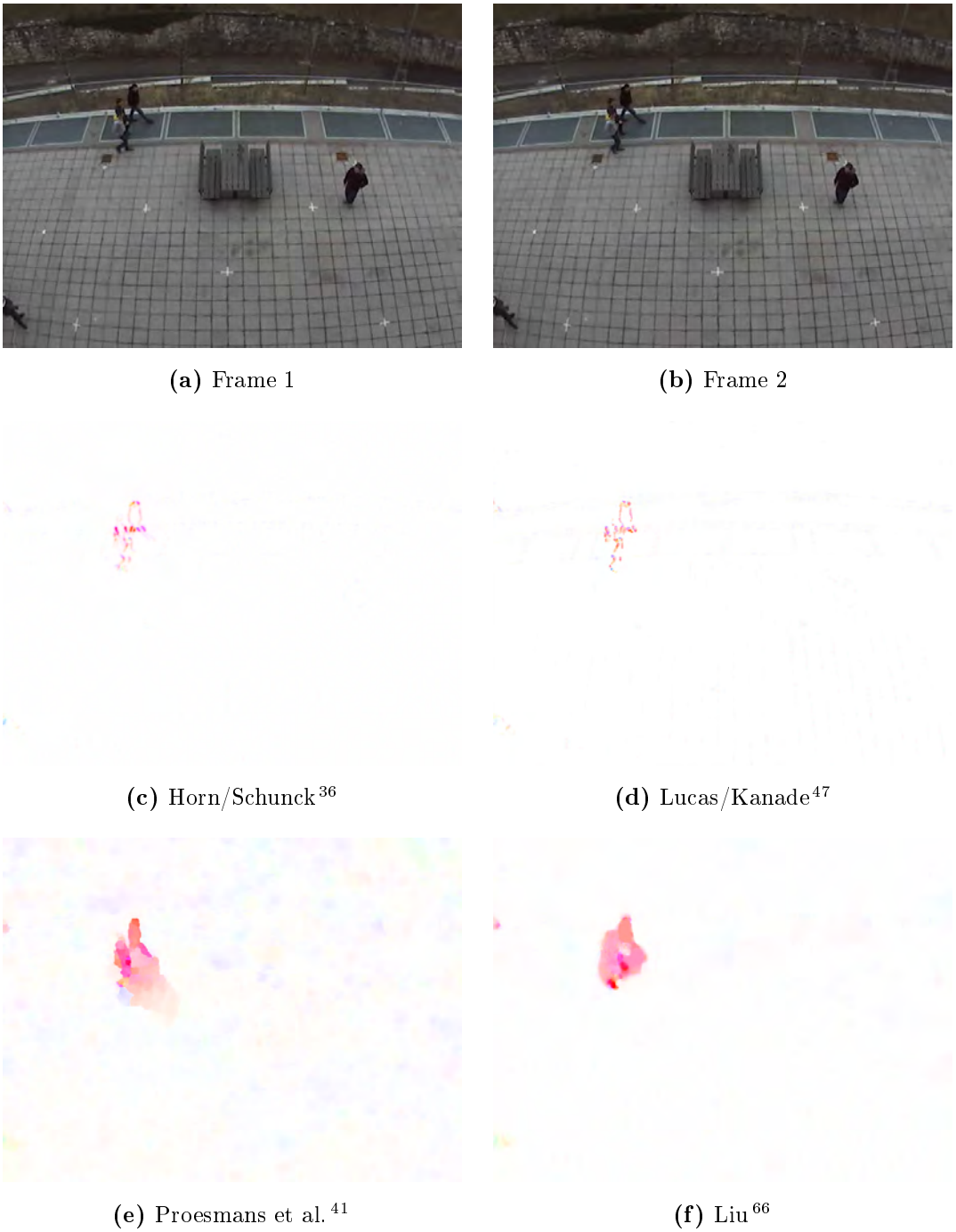
Los otros dos modelos que hemos encontrado han sido el de Proesmans et al.[41] (`http://www.mathworks.es/matlabcentral/fileexchange/40576-proesmans`) y el que desarrolló Liu[66] para su tesis doctoral (`http://people.csail.mit.edu/celiu/OpticalFlow/`).

En la figura 2.9 hemos incluido dos fotogramas consecutivos de uno de los vídeos que hemos usado en nuestros experimentos (*'vigilancia2'*), junto con el resultado del flujo óptico (usando el código de color indicado por Baker et al.[44]) de cada uno de estos cuatro algoritmos.

Como podemos ver, las dos implementaciones que ya vienen incluidas en Matlab apenas realizan un suavizado del flujo (2.9c y 2.9d). Por este motivo, solo detectan bien el movimiento en las regiones cercanas a los bordes de los objetos que se mueven (como el grupo que anda en la zona superior izquierda). Sin embargo, en las áreas con poca textura o sin cambios de iluminación, como en la ropa de estas tres personas, es incapaz de estimar correctamente su velocidad.

La salida producida por los algoritmos de Proesmans et al.[41] y Liu[66] es parecida. No obstante, tras realizar pruebas con diferentes fotogramas y vídeos, observamos que, igual que ocurre en la figura 2.9, el modelo de Proesmans et al.[41] (2.9e) suele ser más ruidoso y estimar peor los movimientos pequeños que el de Liu[66] (2.9f).

Por ejemplo, aunque en la figura no se vea con detalle por el tamaño de las imágenes, hay una persona que entra en escena por la esquina superior izquierda a la misma velocidad que el grupo. Si nos fijamos en los resultados de ambos algoritmos, podemos observar que Liu[66] detecta ese movimiento con mayor claridad -en el código de colores[44], la saturación es proporcional a la velocidad del objeto-.



(a) Frame 1 **(b)** Frame 2

(c) Horn/Schunck[36] **(d)** Lucas/Kanade[47]

(e) Proesmans et al.[41] **(f)** Liu[66]

Figura 2.9. Resultados de diversos algoritmos para el cálculo del flujo óptico. Las imágenes (a) y (b) corresponden a los fotogramas 99 y 100 del vídeo '*vigilancia2*'. Entre estos dos frames, cuatro algoritmos distintos de flujo óptico han sido utilizados para comparar su estimación de movimiento (c, d, e y f). Como podemos ver, (c) y (d) solo detectan correctamente el movimiento de los bordes de los objetos que se desplazan. Los algoritmos (e) y (f) se acercan mucho a la realidad, siendo la principal diferencia entre ellos que (e) introduce más ruido (detecta pequeños movimientos en zonas estáticas).

2.2.2. Magnitudes extraídas

Ahora que ya hemos elegido un algoritmo que estima el movimiento entre fotogramas razonablemente bien, queremos explicar brevemente las dos magnitudes que hemos extraído del flujo óptico para nuestro modelo de saliencia espacio-temporal.

En primer lugar, parece lógico utilizar el resultado del flujo óptico directamente, es decir, la velocidad. Cuando observamos una escena en la que todo se encuentra parado excepto un objeto, siempre tendemos a prestarle más atención que al resto.

Pasado un tiempo, cuando nuestro cerebro aproxima la trayectoria seguida por el objeto a un modelo que le resulta sencillo de predecir -movimiento rectilíneo uniforme, movimiento circular, etc.- ese objeto deja de sobresalir tanto y nos fijamos en otras partes de la escena -generalmente guiados por su saliencia estática.

Para tratar de incluir este complejo mecanismo de una manera sencilla en nuestro modelo, hemos extraído una segunda magnitud del flujo óptico: la aceleración. Aunque somos conscientes de que el funcionamiento real de nuestro sistema de atención visual es más complejo que fijarse en la aceleración de los cuerpos, consideramos que es una buena aproximación.

¿Pero cómo calculamos la aceleración para cada píxel? Sin duda, hay numerosas formas de estimar la aceleración a partir de la velocidad. Aunque como líneas futuras de investigación (sección 4.3) hemos planteado realizar un filtrado que tenga en cuenta varios fotogramas a la vez -por ejemplo, el último segundo de vídeo-, para este trabajo hemos decidido simplificar el procedimiento y considerar el mínimo número de frames posibles: tres.

Gracias a la famosa ecuación 2.1, en la que $\vec{a}$ es la aceleración, $\vec{v}$ la velocidad y t el tiempo

$$\vec{a} = \frac{\partial \vec{v}}{\partial t}, \tag{2.1}$$

y utilizando la notación $\vec{v_i}\,(x, y)$ para referirnos a la velocidad de un fotograma i en las coordenadas $\{x, y\}$, podemos estimar la aceleración en cada píxel restando las velocidades del mismo punto de un objeto en dos fotogramas consecutivos:

$$\vec{a_i}\,(x_i,\ y_i) = \frac{1}{\partial t}\big[\vec{v_i}\,(x_i,\ y_i) - \vec{v}_{i-1}\,(x_{i-1},\ y_{i-1})\,\big] \tag{2.2}$$

Esto quiere decir que el problema no es tan sencillo como parecía. Para poder restar $\vec{v_i}$ y $\vec{v}_{i-1}$, necesitamos conocer la posición del objeto en ambos fotogramas (que a menos que se encuentre parado, no será la misma).

Sin embargo, al disponer de los vectores de movimiento, deja de ser tan complicado. La técnica conocida como *compensación de movimiento* es capaz de estimar cómo era un fotograma i-1 a partir del fotograma i y el flujo óptico entre ambos.

Para ello, a partir de la definición de velocidad (y considerando $\partial t = 1$):

$$\vec{v_i}\,(x_i,\ y_i) = \frac{\partial \vec{x}}{\partial t} = \{x_i,\ y_i\} - \{x_{i-1},\ y_{i-1}\}, \tag{2.3}$$

despeja la posición del objeto en el frame anterior obteniendo:

$$\{x_{i-1},\ y_{i-1}\} = \{x_i,\ y_i\} - \vec{v_i}\,(x_i,\ y_i) \tag{2.4}$$

Como ejemplo, en la figura 2.10, el frame 2 (2.10b) ha sido compensado usando esta técnica para producir la imagen 2.10c. Si las viésemos a gran escala, las subfiguras (a) y (c) solo presentan mínimas diferencias donde hay oclusiones (ya que exclusivamente con la información de un fotograma no se puede predecir lo que hay detrás de los objetos en primer plano).

Basándonos en el mismo principio, podemos "compensar la velocidad" y obtener la aceleración para cualquier píxel combinando las ecuaciones 2.2 y 2.4:

$$\vec{a_i}\,(x,\ y) = \frac{1}{\partial t}\big[\vec{v_i}\,(x,\ y) - \vec{v}_{i-1}\big(\{x,\ y\} - \vec{v_i}\,(x,\ y)\,\big)\big] \tag{2.5}$$

Para explicar mejor este proceso, hemos creado la ilustración de la figura 2.11, en la que mostramos cómo calcular la aceleración del punto rojo en el tercer fotograma: $\vec{a_3}\,(7,3)$. Veamos detalladamente los pasos que hemos seguido.

La primera fila de la figura (a, b y c) representa tres fotogramas consecutivos de un vídeo. Para simplificar el ejemplo, la secuencia consta de un fondo azul fijo y una partícula roja que se desplaza de izquierda a derecha. Hemos escrito un número sobre el punto para identificar la posición que ocupaba en cada frame.

Después de aplicar el algoritmo de flujo óptico entre los fotogramas 1-2 y 2-3, obtendríamos algo similar a las imágenes (d) y (e) respectivamente. Como entre los dos primeros frames la partícula se ha movido 2 píxeles hacia la derecha, $\vec{v_2}$ valdrá 0 para cualquier coordenada excepto donde se sitúa el punto rojo: $\vec{v_2}\,(4,3) = \{2,0\}$. Igualmente, para el tercer fotograma:

$$\vec{v_i}\,(x,\ y) = \vec{v_3}\,(7,\ 3) = \{3,\ 0\} \tag{2.6}$$

Con esto ya tendríamos el primer término de la ecuación 2.5. Para obtener el segundo término, debemos evaluar $\vec{v_2}$ en la posición en la que se encontraba la partícula roja en el segundo fotograma:

$$\vec{v_2}\big(\{7,\ 3\} - \vec{v_3}\,(7,\ 3)\,\big) = \vec{v_2}\big(\{7,\ 3\} - \{3,\ 0\}\big) = \vec{v_2}\,(4,\ 3) = \{2,\ 0\} \tag{2.7}$$

Finalmente, combinando ambos términos como indica la ecuación 2.5 obtendríamos el valor de la aceleración del punto rojo en el tercer frame:

$$\vec{a_3}\,(7,\ 3) = \frac{1}{\partial t}\big[\vec{v_3}\,(7,\ 3) - \vec{v_2}\,(4,\ 3)\,\big] = \frac{1}{\partial t}\big[\{3,\ 0\} - \{2,\ 0\}\big] = \frac{1}{\partial t}\{1,\ 0\} \tag{2.8}$$

(a) Frame 1 **(b)** Frame 2 **(c)** Frame 2 compensado

Figura 2.10. Ejemplo de la compensación de movimiento. A partir del segundo fotograma (b) y los vectores de movimiento, se puede *reconstruir* el fotograma anterior (c).

Por último, queríamos añadir que para el modelo de saliencia espacio-temporal solo hemos empleado el módulo de la velocidad y la aceleración, ya que consideramos que la dirección de estas dos magnitudes no influye en nuestro sistema de atención visual a la hora de centrarse en una región u otra.

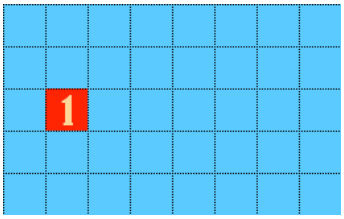
2.2.3. Resultados y ejemplos

Para concluir la sección sobre la estimación de movimiento, hemos querido incluir las figuras 2.12 y 2.13, que muestran la velocidad y aceleración obtenidas para dos secuencias de ejemplo. Nótese que de la misma manera que la compensación de movimiento tiene pequeños fallos cuando se dan oclusiones, la aceleración calculada por este método sufre los mismos problemas.

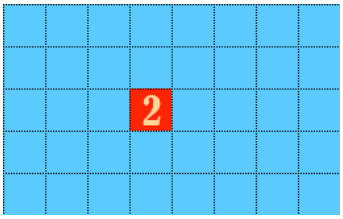
Debido a cuestiones de tamaño, hemos optado por no incluir la imagen de los dos fotogramas anteriores al de las medidas -las diferencias entre frames consecutivos son inapreciables a esta escala-. En su lugar, hemos indicado el número de fotograma y el vídeo al que corresponden los resultados para que puedan identificarse posteriormente.

En el ejemplo de '*vigilancia1*', las dos personas en el centro de la imagen simulan pegarse. Se aproximan rápido, pero una vez están cerca, se frenan. Por este motivo, podemos observar cómo la velocidad de ambos va en dirección de acercamiento, mientras que sus aceleraciones tienen signo contrario (se están frenando).

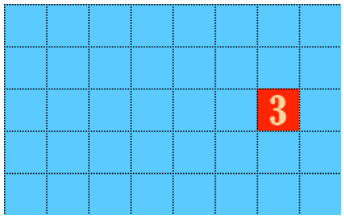
En el segundo ejemplo, se puede observar la buena precisión del algoritmo de flujo óptico, que detecta tanto el movimiento del grupo en el centro de la imagen, como el desplazamiento de la persona que apenas se distingue en la zona superior. Por su parte, la aceleración es casi nula en la totalidad de la imagen, exceptuando en los bordes de los que andan (debido a la oclusión) y en el pie de la persona de gris, que acaba de apoyar en el suelo y por tanto está decelerando su movimiento de avance.



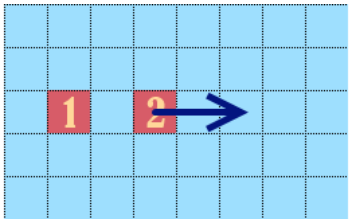
(a) Frame 1



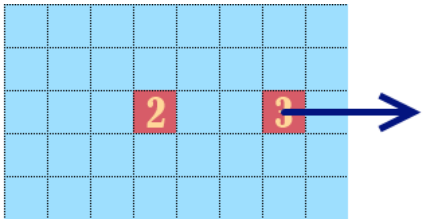
(b) Frame 2



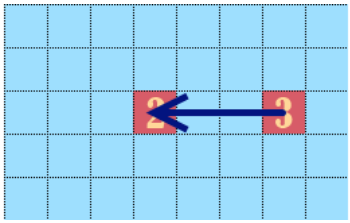
(c) Frame 3



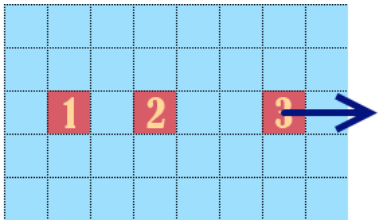
(d) $\vec{v_2}\,(4,\ 3)$: Movimiento entre 1 y 2



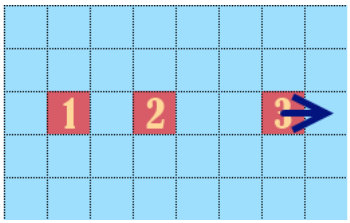
(e) $\vec{v_3}\,(7,\ 3)$: Movimiento entre 2 y 3



(f) $-\vec{v_3}\,(7,\ 3)$

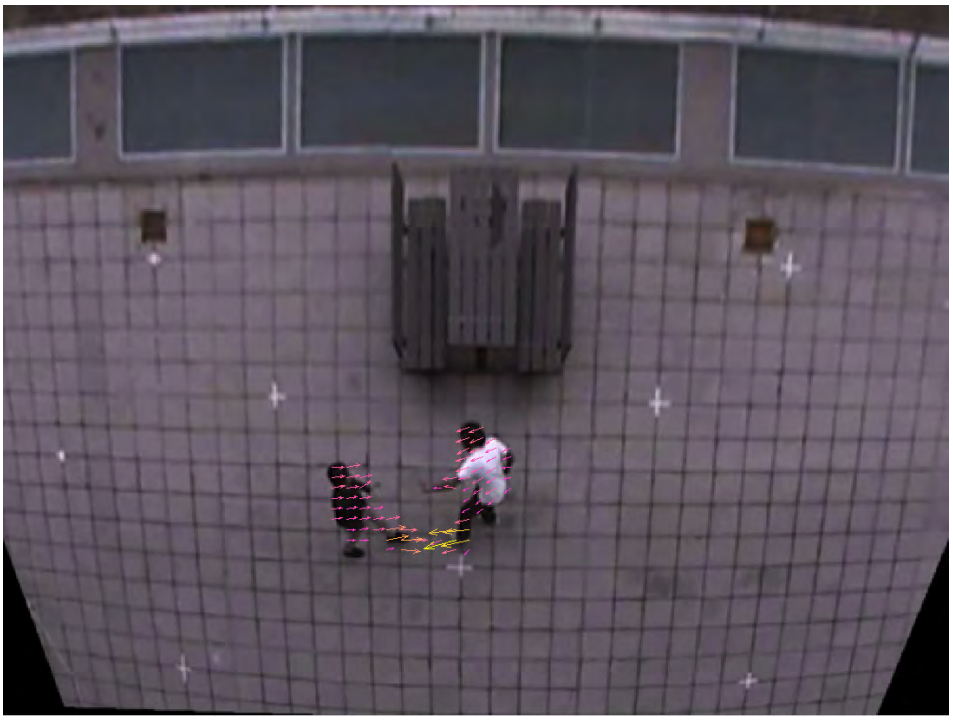


(g) $\vec{v_2}\,(4,\ 3)$ (colocado sobre $\{7,\ 3\}$)

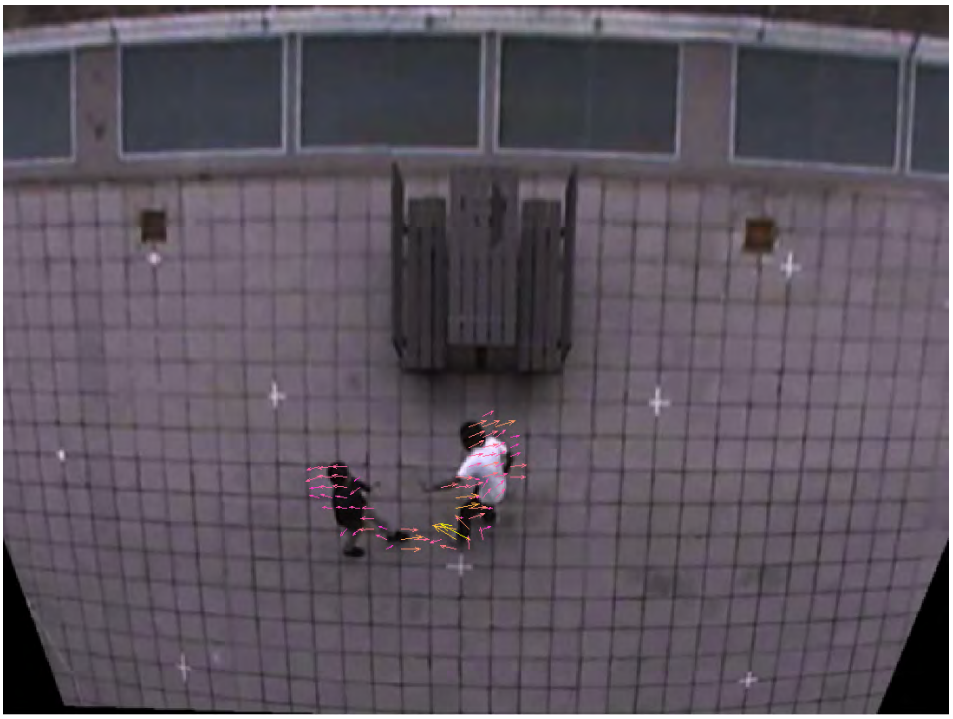


(h) $\vec{a_3}\,(7,\ 3)$: Aceleración entre 1 y 3

Figura 2.11. Ejemplo sencillo para explicar el cálculo de la aceleración.
En la primera fila, (a), (b) y (c) simbolizan tres fotogramas de una secuencia de vídeo. Debajo, (d) y (e) muestran la velocidad de cada píxel en los frames 2 y 3 respectivamente. (f) representa la ecuación 2.4: el punto rojo, que en el frame 3 estaba en la posición $\{7,3\}$, se encontraba en el frame anterior donde apunta la flecha azul: $\{7,3\} - \vec{v_3}\,(7,3) = \{4,3\}$. Gracias a esto, podemos obtener "$\vec{v_2}$ compensada" (g), que no es más que colocar en el frame 3 los vectores de velocidad que tenía cada objeto en el frame 2. Por último, según indica la ecuación 2.5, la aceleración en el frame 3 se obtiene como $(e) - (g) = (h)$: $\{3,\ 0\} - \{2,\ 0\} = \{1,\ 0\} = \vec{a_3}\,(7,\ 3)$.



(a) Velocidad en el frame 1759 de '*vigilancia1*'



(b) Aceleración en el frame 1759 de '*vigilancia1*'

Figura 2.12. Resultados de velocidad y aceleración en el vídeo '*vigilancia1*'. En el vídeo, las dos personas se acercan pero cada vez más lento. Por eso, su velocidad (a) apunta hacia la otra persona, mientras que su aceleración (b) tiene signo contrario.

(a) Velocidad en el frame 167 de '*vigilancia2*'

(b) Aceleración en el frame 167 de '*vigilancia2*'

Figura 2.13. Resultados de velocidad y aceleración en el vídeo '*vigilancia2*'. En el vídeo, el grupo del centro y la persona de arriba a la izquierda se mueven a velocidad (a) constante. Por eso, su aceleración (b) es prácticamente nula.

2.3. Segmentación de los fotogramas

Como ya comentamos anteriormente en la sección 1.4, decidimos incluir un paso de segmentación del fotograma de entrada por dos motivos principales. En primer lugar, a lo largo de este capítulo hemos podido observar que ni el algoritmo de saliencia espacial ni los cálculos del flujo óptico son perfectos.

Por eso, hemos considerado oportuno combinar estos mapas con una segmentación externa, de forma que todos los píxeles de un segmento compartan el mismo valor de cada característica. Estableciendo dicho valor a la media de cada característica a lo largo del segmento, hemos conseguido suavizar los mapas manteniendo las fronteras entre objetos, y atenuar los efectos de las oclusiones en el cálculo de la aceleración.

El segundo motivo por el que hemos creído necesario usar una segmentación es que el escaso *ground truth* del que disponemos fue tomado con un *eye-tracker* de poca precisión. En consecuencia, necesitábamos alguna herramienta que nos permitiese determinar los objetos a los que el sujeto estaba mirando.

Debido a la subjetividad de la segmentación, ya comentamos que no hay bases de datos públicas que permitan ordenar los algoritmos de mejor a peor. Tras investigar sobre diferentes implementaciones en Matlab de este tipo de herramientas, encontramos el software de segmentación de EDISON[53] (Edge Detection and Image SegmentatiON: `http://coewww.rutgers.edu/riul/research/code/EDISON/`), que resultó ser muy sencillo de utilizar y configurar.

Como la librería cumple suficientemente bien con la función para la que la necesitamos y nos basta con ser capaces de identificar zonas similares como un único bloque, no hemos incluido un apartado de comparación entre esta herramienta y otras similares como sí hicimos para la saliencia estática y el flujo óptico.

Para concluir con esta sección, hemos añadido las figuras 2.14, 2.15 y 2.16. En la primera mostramos los resultados de la segmentación que produce EDISON con varias imágenes de prueba. En la segunda y la tercera, se puede ver cómo quedarían los mapas de saliencia estática, velocidad y aceleración si a cada segmento le asignamos el valor medio de todos sus píxeles.

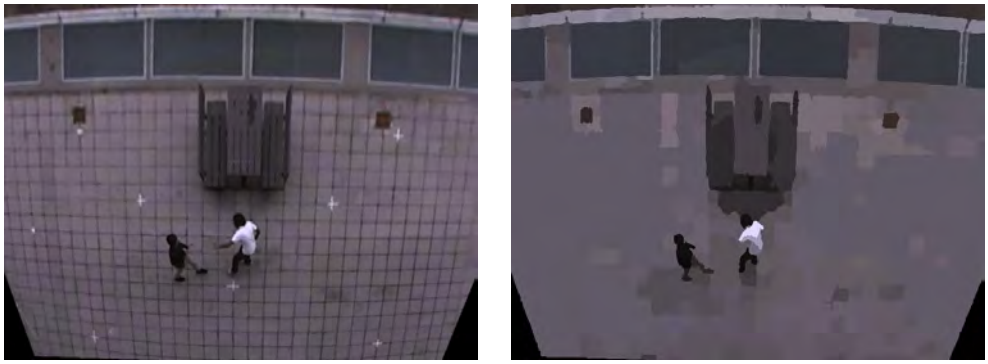
Como ya hemos mencionado, aunque en las imágenes no se aprecien ventajas aparentes, la segmentación nos será de gran ayuda para los experimentos que vamos a detallar en el siguiente capítulo.

(a) Imagen original **(b)** Segmentación

(c) Imagen original **(d)** Segmentación

(e) Imagen original **(f)** Segmentación

Figura 2.14. Segmentación de varias imágenes de prueba con EDISON[53].



(a) Imagen original

(b) Segmentación

(c) Saliencia original

(d) Saliencia + Segmentación

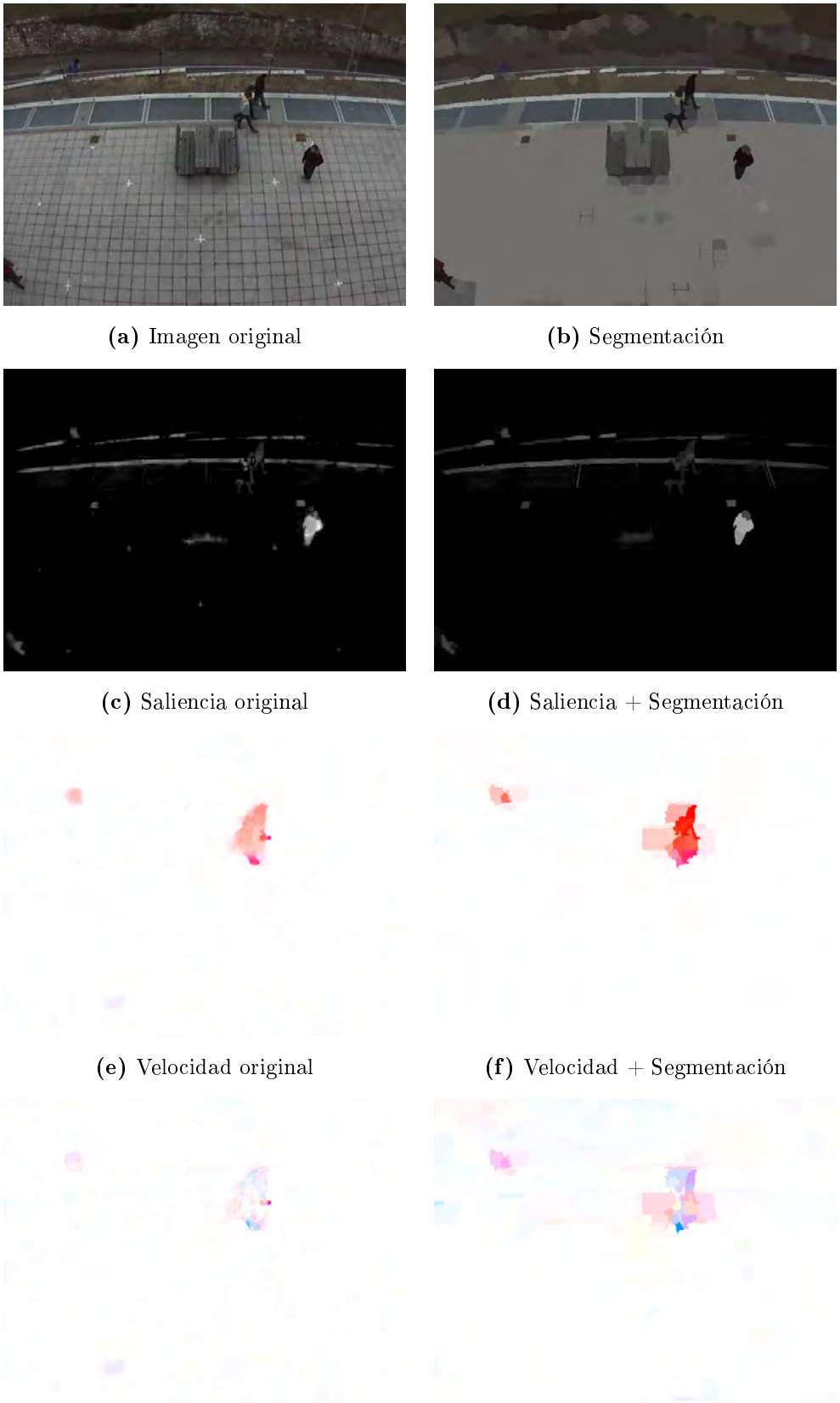
(e) Velocidad original

(f) Velocidad + Segmentación

(g) Aceleración original

(h) Aceleración + Segmentación

Figura 2.15. Resultados tras la segmentación del frame 1759 de '*vigilancia1*'.



(a) Imagen original

(b) Segmentación

(c) Saliencia original

(d) Saliencia + Segmentación

(e) Velocidad original

(f) Velocidad + Segmentación

(g) Aceleración original

(h) Aceleración + Segmentación

Figura 2.16. Resultados tras la segmentación del frame 167 de '*vigilancia2*'.

Capítulo 3
Experimentos

En el capítulo anterior hemos descrito los mapas de características que vamos a usar para nuestro modelo de saliencia espacio-temporal y el procedimiento y las herramientas para obtener cada uno de ellos.

En este capítulo, vamos a explicar cómo hemos entrenado a una red neuronal para que estime las ponderaciones más óptimas de cada mapa de características, de forma que al combinarlos obtengamos la mejor estimación de saliencia posible.

Para ello, comenzaremos hablando del escaso *ground truth* del que disponemos, así como de los distintos problemas que nos han ido surgiendo. Después, detallaremos cómo hemos realizado el proceso de entrenamiento de la red neuronal y justificaremos nuestra elección final con ejemplos.

Por último, incluiremos varias figuras demostrando el funcionamiento de nuestro modelo ante diversas secuencias, y realizaremos una comparación entre nuestra salida y la que produce el algoritmo de saliencia estática del `SaliencyToolBox`.

3.1. Obtención del *ground truth*

Como mencionamos en la motivación del proyecto, la iniciativa de desarrollar un modelo de atención visual para secuencias de vídeo estuvo motivada por la disposición de los datos recogidos por un *eye-tracker* como parte de los experimentos que llevó a cabo Muratov[1] para su tesis doctoral.

Sin embargo, el dispositivo utilizado para detectar el píxel de la pantalla al que estaban mirando en cada instante los individuos que participaron en los experimentos no estaba diseñado para tener alta precisión.

Más concretamente, el *eye-tracker* que empleó Muratov fue originalmente desarrollado como una herramienta *low-cost* que pudiese servir a personas paralíticas para comunicarse con sus ojos[1], gracias al proyecto de investigación de Armanini y Conci[67].

Debido a esta finalidad original, este barato dispositivo no ofrece una precisión muy elevada. Por este motivo, hemos encontrado numerosas dificultades a la hora de obtener un *ground truth* fiable, que nos indicase exactamente qué objetos del vídeo se miraban en cada momento.

Los datos a los que hemos tenido acceso se componen de cuatro vídeos y 35 archivos ".xml" en los que se almacena el píxel al que miraban los individuos en cada fotograma. Sin embargo, dos de los vídeos apenas duran unos segundos y algo más de la mitad de los experimentos corresponden a dichos vídeos o a secuencias de las que no disponemos.

Por tanto, nuestra base de datos se reduce a 2 vídeos y 12 ".xml". Estas secuencias son '*vigilancia1*'[2] y la que sería '*vigilancia3*', pero que como explicaremos dentro de unos párrafos tuvimos que descartar debido a los errores en las medidas.

Utilizando un pequeño script que incluyeron con los archivos, pudimos reproducir los experimentos y acceder a las posiciones en las que se fijaron los seis individuos que participaron en la elaboración de la base de datos.

En la figura 3.1 se muestran tres fotogramas del vídeo '*vigilancia1*' con un pequeño círculo blanco superpuesto indicando la posición de la mirada de uno de los individuos.

Como podemos ver, el error del *eye-tracker* es grande. El radio del círculo blanco de la figura 3.1 mide 25 píxeles y aun así en ninguno de estos tres fotogramas toca mínimamente a las dos personas que aparecen.

(a) Fotograma 250 **(b)** Fotograma 1000 **(c)** Fotograma 3000

Figura 3.1. Posición de la mirada del primer individuo en el vídeo *vigilancia1*.

[1] Más información sobre el *eye-tracker* utilizado y el proyecto original se puede encontrar en: `http://www.xtensa.it/index.html` y `http://www.otbfoundation.org/projects/eye-assist`.

[2] En el apéndice A se puede encontrar toda la información relativa a los vídeos utilizados.

Además, después de observar los resultados recopilados para todos los frames, hemos notado que la frecuencia de muestreo del aparato es demasiado baja. Es decir, ante un movimiento de ojos constante, los datos almacenados simulan un desplazamiento "a trompicones".

Por tanto, cuando las personas del vídeo andan a velocidad constante y el individuo las sigue con la mirada, lo que se registra como *ground truth* es un frame en el que el círculo blanco se encuentra encima de la persona, seguido de unos diez fotogramas en los que el punto permanece quieto mientras la persona ya no está allí.

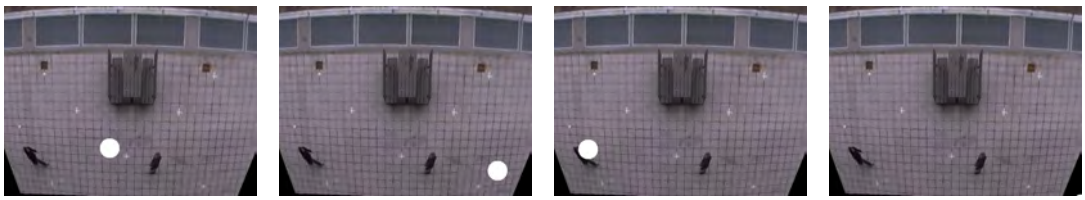
Como decíamos, esta baja frecuencia de muestreo y la mala calidad del *eye-tracker*, que en determinados experimentos incluso dejaba de seguir la mirada del individuo durante algunos minutos, nos han supuesto muchos problemas.

Para tratar de minimizar los errores, tomamos dos medidas al respecto. En primer lugar, analizamos los resultados de los doce experimentos y nos quedamos con el único que parecía ser muy preciso -cuando las personas se movían, el círculo blanco se situaba encima de ellas aproximadamente el 90 % del tiempo.

En las figuras 3.2 y 3.3 mostramos como ejemplo una comparativa de la mirada de cuatro de los seis individuos recogida por el *eye-tracker* (parece que el dispositivo se desconfiguró pasados unos segundos del inicio del vídeo en los otros dos experimentos).

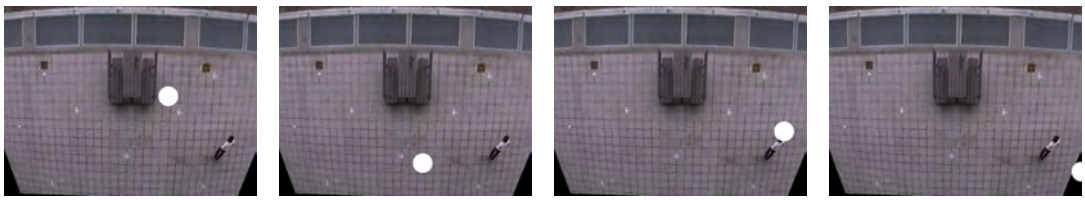
Además, como hemos anticipado anteriormente, los resultados con el segundo vídeo son muy malos. Es posible que el *eye-tracker* no se configurase correctamente, ya que es bastante sensible a la iluminación del ambiente. De todas maneras, los datos recogidos para esta secuencia no iban a generar un buen modelo de saliencia espacio-temporal, por lo que decidimos descartarlos y quedarnos únicamente con el individuo 3 (figuras 3.2c y 3.3c).

Somos conscientes de que los datos que podamos extraer de un único experimento no van a dar lugar a un modelo suficientemente robusto y generalizable para otros tipos de vídeos.



(a) Individuo 1 **(b)** Individuo 2 **(c)** Individuo 3 **(d)** Individuo 4

Figura 3.2. Comparación de los datos obtenidos para el fr. 1000 de *vigilancia1*.



(a) Individuo 1 **(b)** Individuo 2 **(c)** Individuo 3 **(d)** Individuo 4

Figura 3.3. Comparación de los datos obtenidos para el fr. 1960 de *vigilancia1*.

Debido a la falta de medios -no disponemos de un *eye-tracker*- y la limitación de tiempo, hemos propuesto obtener una base de datos mucho más grande como una de las líneas futuras de este proyecto (sección 4.3). No obstante, el procedimiento que vamos a describir es fácilmente modificable para incluir los datos de más de un experimento.

Para obtener el *ground truth* a partir de los datos del *eye-tracker* para el individuo 3, el método que hemos seguido se explica a continuación (ilustrado con el diagrama de flujo de la figura 3.4).

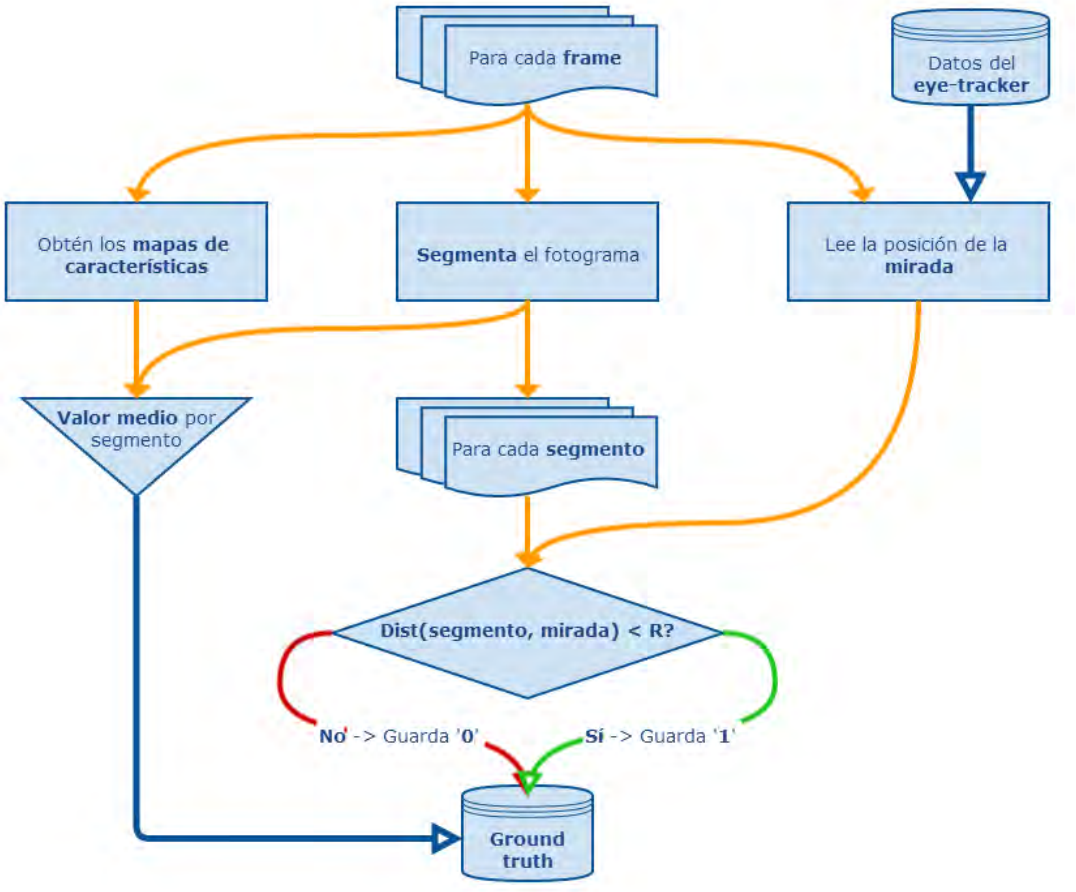


Figura 3.4. Proceso de extracción del *ground truth* de los datos del *eye-tracker*.

Para cada fotograma, obtenemos los tres mapas de características como se detalló en el capítulo 2, segmentamos la imagen y leemos el píxel al que estaba mirando el individuo.

Posteriormente, calculamos la saliencia estática, velocidad y aceleración medias de cada segmento y determinamos a cuáles de ellos estaba mirando el individuo. Para ello, en función de un parámetro configurable R, establecemos como salientes (valor 1) todos aquellos que tengan algún píxel dentro del círculo de radio R centrado en la posición que marca el *eye-tracker*, mientras que el resto de segmentos se consideran no salientes (0).

Por último, guardamos en nuestra base de datos los valores de las entradas (saliencia espacial, velocidad y aceleración medias) y la salida deseada (saliente o no saliente) para cada segmento.

Con más experimentos, bastaría con ampliar la escala de 0-1 a 0-n (siendo n el número de experimentos), de forma que cada segmento tendría asignado un número que representaría la cantidad de individuos que lo estaban mirando en ese fotograma. Para este caso, la figura 3.5 muestra cómo quedaría el diagrama de flujo.

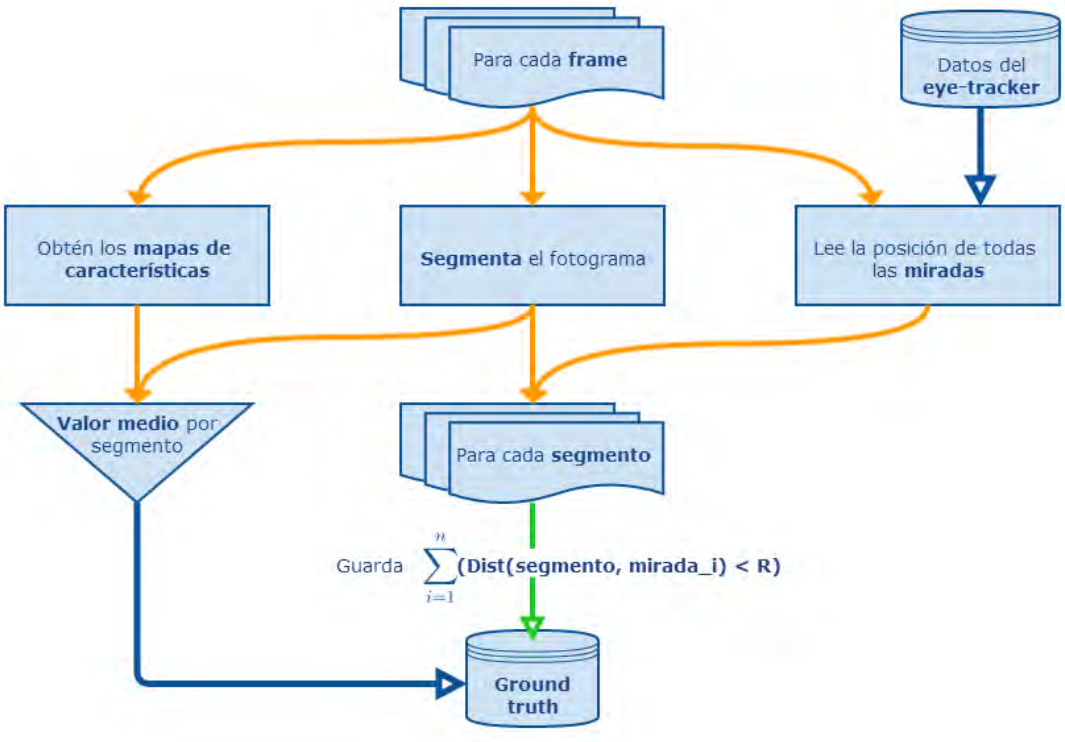


Figura 3.5. Proceso de obtención del *ground truth* con datos de varias personas.

Sin embargo, al emplear este procedimiento nos dimos cuenta de varios problemas. Para empezar, estábamos incluyendo un montón de *outliers* como falsos positivos. Como podemos observar en la figura 3.6, marcando como salientes todos los segmentos que tuviesen algún píxel dentro del "círculo blanco" se consigue que la gran parte del suelo -en absoluto saliente- sea considerada como llamativa.

Pero además, necesitábamos usar un radio R mucho más grande, ya que en numerosas ocasiones, debido al error del *eye-tracker*, solo una pequeña porción de la persona a la que miraba el individuo (como la cabeza o un brazo) era detectada como positiva, marcando el resto del cuerpo como negativo (figura 3.6b).

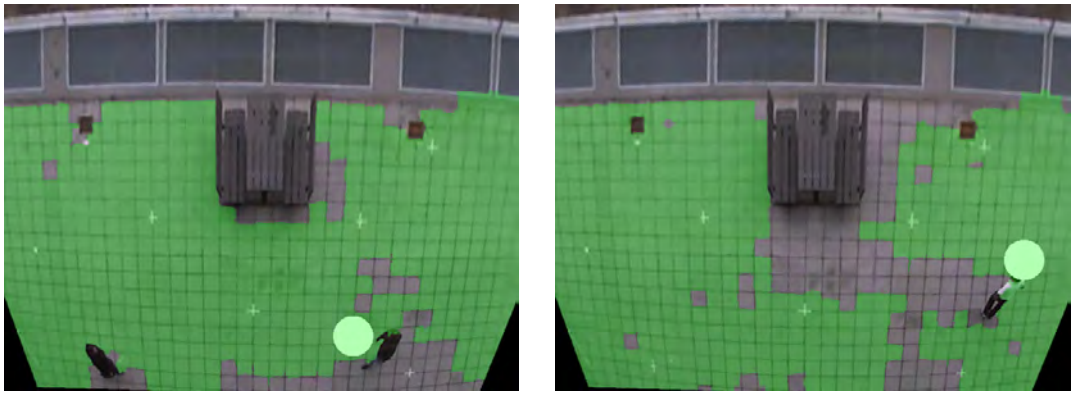
También nos dimos cuenta de que, al utilizar los datos de la mirada de una única persona, solo podíamos atender a un objeto saliente, incluso si la imagen contenía varios. En numerosas ocasiones, por falta de datos, estábamos incluyendo en nuestra base de datos bastantes falsos negativos.

Por ejemplo, en la figura 3.6a el individuo solo puede mirar a una persona, así que la otra se almacena como no saliente. Sin embargo, es probable que para otro espectador la persona de la izquierda sea más llamativa, por lo que entrenar una red neuronal con datos *a priori* contradictorios puede resultar muy negativo.

Para tratar de solucionar estos problemas se nos ocurrió la modificación que se muestra en la figura 3.7.

El método propuesto, como se puede observar en el diagrama, consiste en utilizar la información de los mapas de saliencia espacial, velocidad y aceleración para intuir si el segmento a identificar es un posible candidato o un *outlier*.

Más concretamente, hemos abierto las dos posibilidades -saliente o no- a cuatro opciones: positivo, posible *outlier* positivo, posible *outlier* negativo y negativo.



(a) Frame 100 **(b)** Frame 1475

Figura 3.6. Ejemplos de los *outliers* incluidos originalmente en el *ground truth*.
Código de color: sobresaltado en verde → saliente; resto → no saliente.

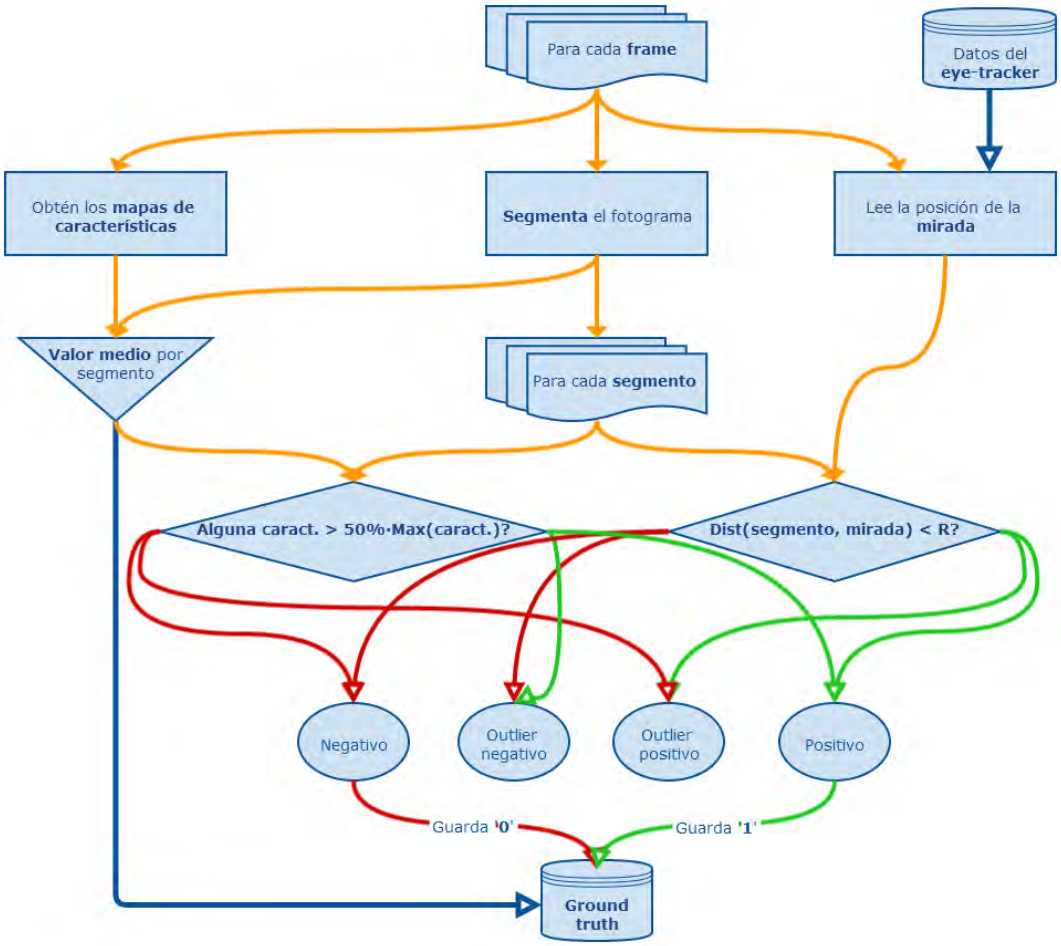
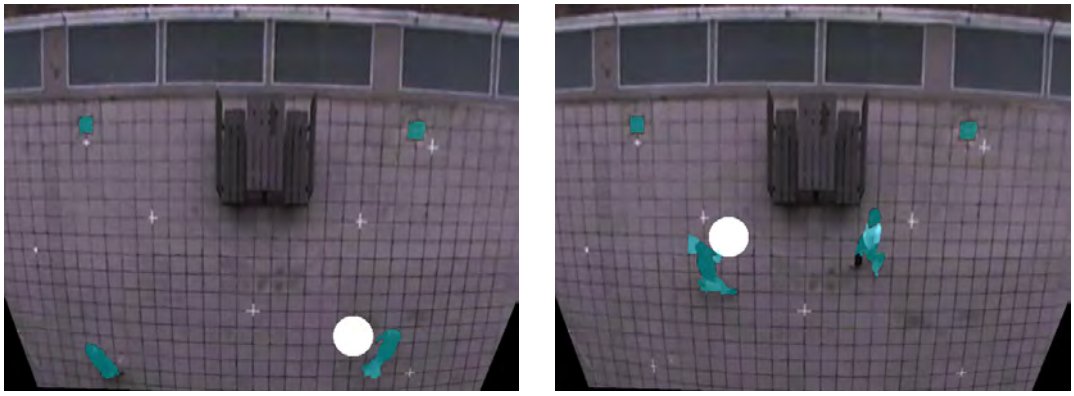


Figura 3.7. Proceso de obtención del *ground truth* mejorado.

Después, hemos descartado las muestras que podrían ser *outliers* y solo hemos incluido en nuestro *ground truth* los segmentos que parecen verdaderos positivos o negativos.

La pregunta ahora es: ¿cómo identificamos un segmento como posible *outlier*? Después de realizar varias pruebas, hemos observado que prácticamente siempre que nos fijamos en un objeto es porque alguna de las tres características destaca sobre el resto (véase la figura 3.8).

Es decir, si analizásemos las características -saliencia estática, velocidad o aceleración- de todos los objetos -o segmentos- de una escena, veríamos que las regiones en las que más nos fijamos se encuentran por encima de la media en alguna de estas tres magnitudes. Por este motivo, hemos introducido el heurístico del 50 % del valor máximo de cada mapa como indicador de posible *outlier*. Debemos recordar que no sería necesario incluir esta medida si dispusiésemos de los resultados de un mayor número de experimentos para el mismo vídeo (figura 3.5).



(a) Frame 100 **(b)** Frame 1475

Figura 3.8. Ejemplos de la detección de posibles *outliers*. Sobresaltado en azul: regiones con saliencia estática, velocidad y/o aceleración por encima del 50 % del máximo.

Al añadir nuestro heurístico, se pueden dar cuatro combinaciones:

a) El segmento tiene algún píxel dentro del "círculo blanco" y una o más características se encuentran por encima de la media. Muy probablemente, esta región será saliente. En consecuencia, guardamos los datos del segmento como *ground truth* positivo.
b) El segmento tiene algún píxel dentro del círculo pero ninguna de sus características destaca. Es posible que nos encontremos ante un *outlier* positivo (como el suelo en la figura 3.6), por lo que ante la duda no lo incluimos en nuestra base de datos.
c) El segmento está fuera del círculo pero alguna de sus características destaca. Similar al caso b), podríamos estar ante un *outlier* negativo (como la segunda persona de la figura 3.6a), así que no almacenamos los datos de este segmento en nuestro *ground truth*.
d) El segmento está fuera del círculo y ninguna de sus características llama la atención. Muy probablemente, esta región no es saliente. Por consiguiente, guardamos los datos del segmento como *ground truth* negativo.

Finalmente, hemos construido dos bases de datos. La primera, que hemos denominado 'GT original', obtenida por el método del diagrama 3.4; y la segunda, 'GT procesado', obtenida por el procedimiento de la figura 3.7.

En el cuadro 3.1 hemos incluido una pequeña comparativa entre ambas bases de datos. Al contrario de lo que pueda parecer a primera vista, 'GT procesado' tiene un porcentaje de segmentos positivos mucho menor, debido a que la cantidad de *outliers* positivos (regiones del suelo) que hemos eliminado es muy superior a la de *outliers* negativos (las dos alcantarillas y la segunda persona).

	'GT original'	**'GT procesado'**
Número total de segmentos	3 254 314	2 998 858
Porcentaje de positivos	5.24 %	0.79 %
Porcentaje de negativos	94.76 %	99.21 %

Cuadro 3.1. Información sobre las bases de datos generadas como *ground truth*.

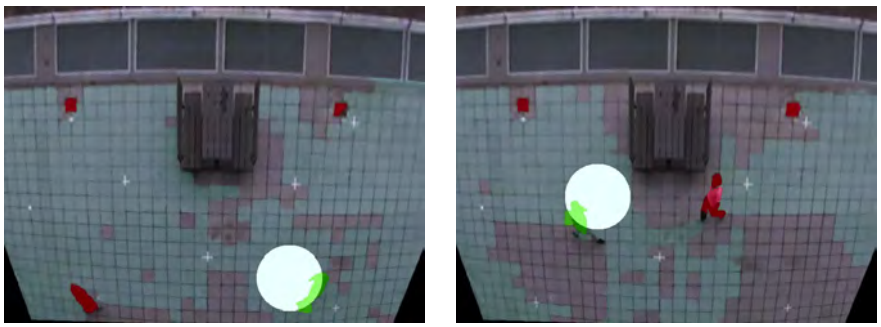
	'GT 50-50'	**'GT 25-75'**	**'GT 10-90'**
Número total de segmentos	47 390	94 780	236 950
Porcentaje de positivos	50 %	25 %	10 %
Porcentaje de negativos	50 %	75 %	90 %

Cuadro 3.2. Información sobre las bases de datos adicionales.

Además, podemos ver cómo ambas bases de datos están muy sesgadas hacia salidas negativas. Para intentar evitar que el aprendizaje máquina nunca produzca valores positivos, hemos diezmado las muestras negativas de 'GT procesado' dando lugar a las tres bases de datos del cuadro 3.2: 'GT 50-50', 'GT 25-75' y 'GT 10-90'.

Para terminar la sección, hemos querido incluir la figura 3.9, que muestra ejemplos de la obtención de la base de datos 'GT procesado'. Nótese que gracias al método de eliminación de posibles *outliers*, hemos podido incrementar el radio del "círculo blanco" y así cubrir por completo las regiones salientes.

Como podemos ver, a diferencia del procedimiento original (figura 3.4) este método no considera la gran parte del suelo como saliente. Igualmente, tampoco incluye en la base de datos ni las alcantarillas ni la segunda persona por no estar seguro de que esos segmentos no sean salientes.



(a) Frame 100 **(b)** Frame 1475

Figura 3.9. Ejemplos de la construcción de la base de datos 'GT procesado'. Código de color: en verde → positivo (añadido); azul claro → posible *outlier* positivo (descartado); rojo → posible *outlier* negativo (descartado); resto → negativo (añadido).

3.2. Entrenamiento de la red neuronal

Una vez generadas estas cinco bases de datos, hemos entrenado a un MLP (Multi-Layer Perceptron) como el de la figura 3.10 para poder combinar los tres mapas de características de nuestro modelo de saliencia espacio-temporal maximizando la tasa de acierto.

Para conseguirlo, hemos utilizado la sencilla interfaz gráfica que ofrece Matlab, y el algoritmo *Scaled conjugate gradient*[63] como se explicó en la sección 1.5, por ser el más rápido y requerir poca memoria RAM.

Como detallaremos en el apéndice A, al sólo disponer de un vídeo con *ground truth* nos hemos visto obligados a dividirlo en dos secuencias: '*vigilancia1_1*', que está formada por alrededor del 80 % inicial del vídeo original, y '*vigilancia1_2*', que consiste en los últimos dos minutos de '*vigilancia1*'.

De este modo, '*vigilancia1_1*' ha sido utilizado únicamente para entrenar al MLP, mientras que '*vigilancia1_2*' nos ha permitido evaluar la efectividad de nuestro modelo y compararla con la salida del `SaliencyToolBox`.

Para el entrenamiento, además de usar las cinco bases de datos que hemos generado utilizando el procedimiento de la sección anterior, hemos realizado pruebas variando algunos parámetros.

Por ejemplo, sabemos que la potencia de una red neuronal como el MLP depende del número de neuronas en su capa oculta. Por tanto, hemos analizado el efecto que tiene en la tasa de acierto variar dicho número. Para ello, hemos realizado pruebas con perceptrones de entre 5 y 30 neuronas en su capa oculta.

Con nuestras bases de datos, hemos observado que incluso 5 neuronas producen un rendimiento muy alto. Además, hemos visto que a partir de las 20 neuronas los resultados no mejoran al incrementar el tamaño de la capa oculta. Como ejemplo, hemos incluido en la figura 3.11 las curvas ROC obtenidas al entrenar un perceptrón de 5 neuronas y otro de 20 con el mismo *ground truth*.

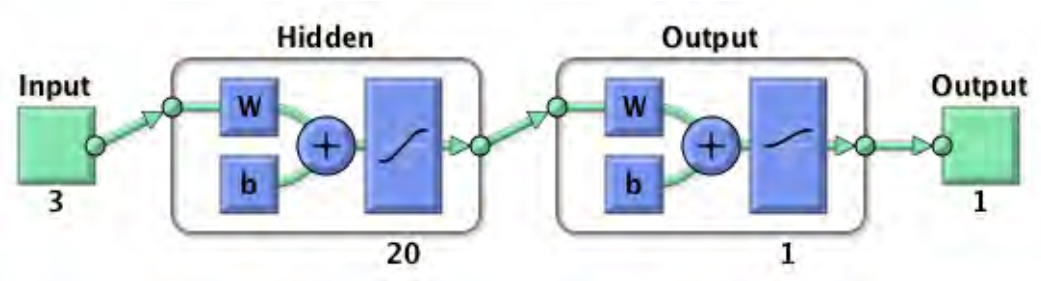
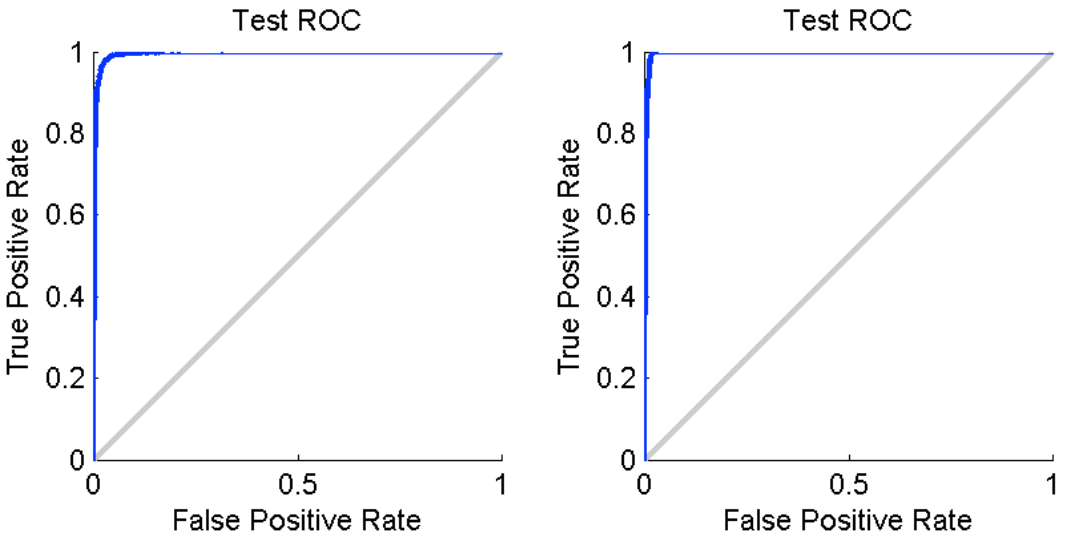


Figura 3.10. Representación del MLP empleado para el aprendizaje máquina.



(a) MLP con 5 neuronas **(b)** MLP con 20 neuronas

Figura 3.11. Curvas ROC de MLPs con distintos tamaños de capa oculta. Hasta cierto punto, incrementar el número de neuronas en la capa oculta de un perceptrón multicapa puede mejorar la potencia de la red, y por tanto su efectividad.

Por otro lado, también hemos analizado el efecto de la normalización del movimiento. Es decir, con cada base de datos, hemos probado a entrenar a la red neuronal de dos formas distintas: pasándole el módulo de la velocidad y aceleración en valor absoluto, y pasándole dichas magnitudes divididas por el valor máximo que alcanzan en ese frame.

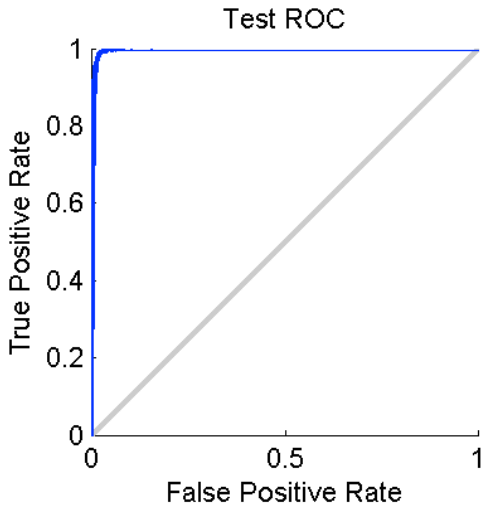
Analizando los datos, hemos observado algo curioso. Fijándonos únicamente en los resultados estadísticos (figura 3.12), las matrices de confusión y las curvas ROC parecen indicar que normalizar el movimiento produce resultados ligeramente mejores.

Sin embargo, cuando vemos la salida de ambos perceptrones en los vídeos, observamos que si normalizamos la velocidad y la aceleración, las regiones detectadas como las más llamativas "parpadean" y varían mucho entre fotogramas consecutivos.

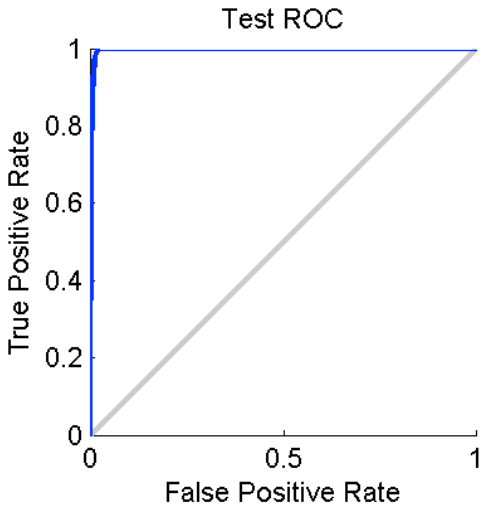
Por el contrario, cuando se utiliza el valor original de ambas magnitudes, las zonas detectadas son mucho más constantes en el tiempo. Para ilustrar este hecho, hemos incluido en las figuras 3.13 y 3.14 las 5 zonas más llamativas en tres fotogramas consecutivos de los vídeos '*vigilancia1_2*' y '*vigilancia2*' respectivamente. Como se puede ver en las imágenes, la salida de las columnas de la izquierda (sin normalización) se mantiene estable en el tiempo, mientras que las columnas de la derecha (normalizando el movimiento) varían rápidamente.

Además, si analizamos detenidamente las estadísticas de la figura 3.12, podemos darnos cuenta de una cosa: la normalización del movimiento tiene una tasa total de acierto superior (99.2 % frente a 99 %) porque cuando no normalizamos, se dan más casos de 'falsos positivos' (6400 frente a 5022). Este hecho, que podría ser interpretado a priori como algo negativo, es el causante de esa estabilidad que notamos en las figuras 3.13 y 3.14.

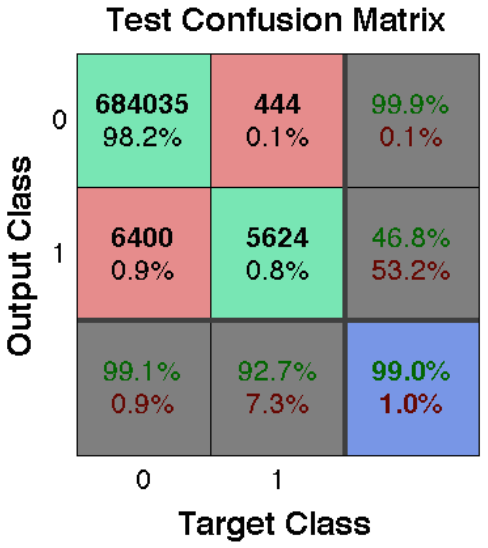
Asimismo, podemos observar que si no normalizamos el movimiento, la cantidad de segmentos que deberían ser salientes y no se identifican como tal es menor (444 frente a 526).



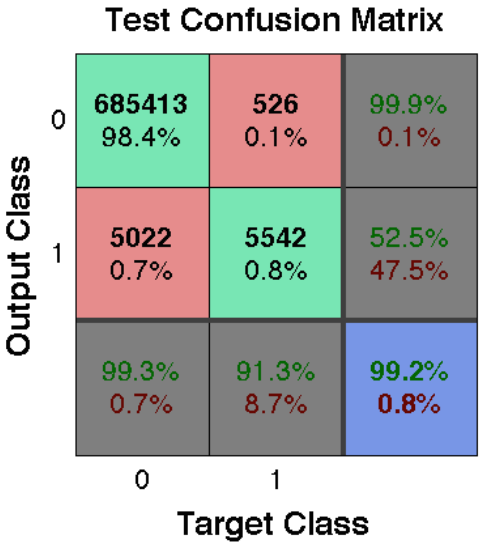
(a) Sin normalización de movimiento



(b) Normalizando velocidad y aceleración

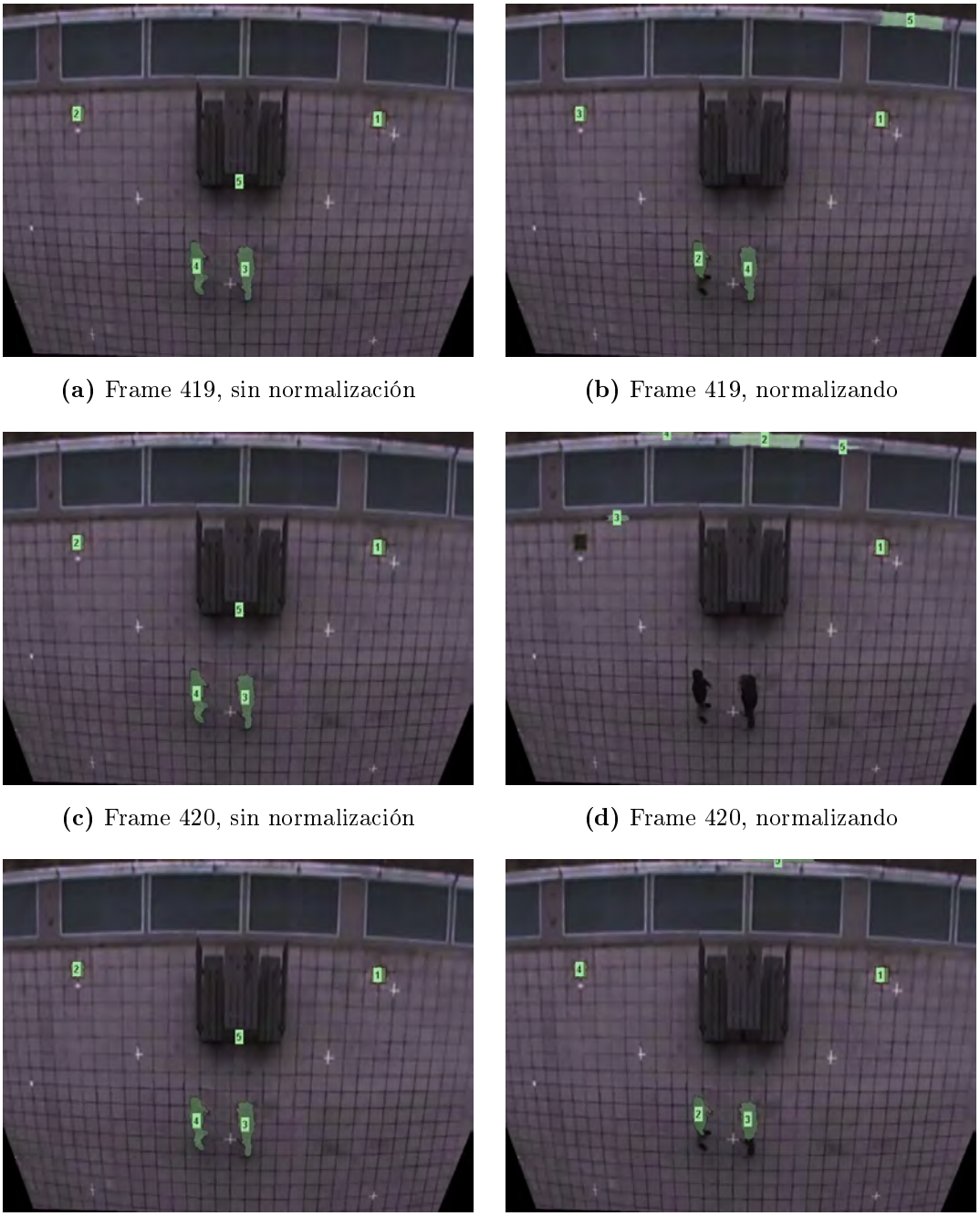


(c) Sin normalización de movimiento



(d) Normalizando velocidad y aceleración

Figura 3.12. Resultados del entrenamiento según la normalización del movimiento.



(a) Frame 419, sin normalización

(b) Frame 419, normalizando

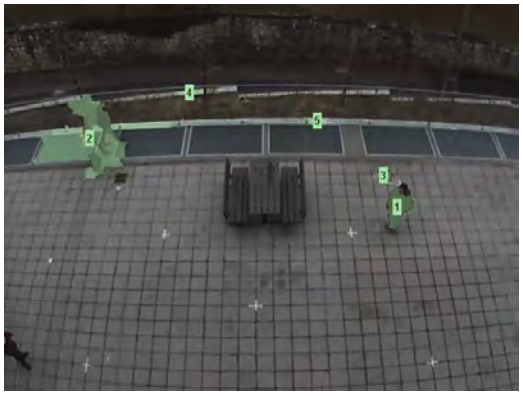
(c) Frame 420, sin normalización

(d) Frame 420, normalizando

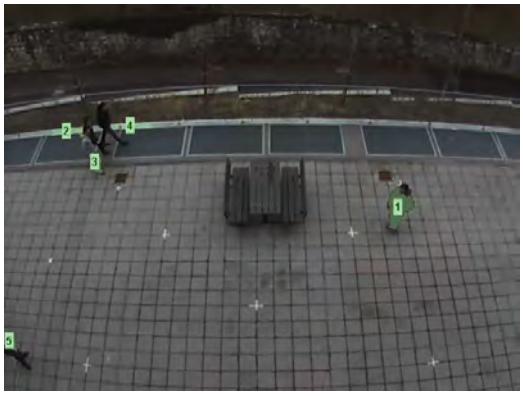
(e) Frame 421, sin normalización

(f) Frame 421, normalizando

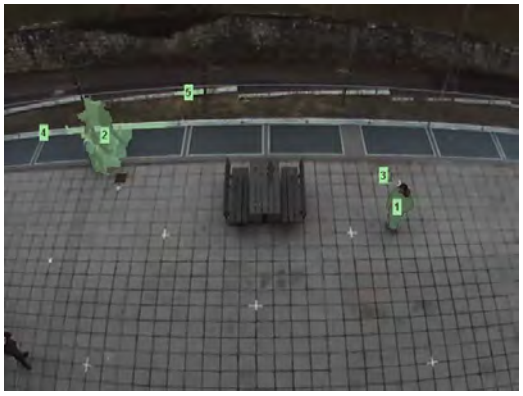
Figura 3.13. Resultados en '*vigilancia1_2*' según la normalización del movimiento. Como podemos observar, cuando dividimos la velocidad y la aceleración por su valor máximo en cada fotograma (b, d y f), cualquier mínima perturbación en algún punto de la imagen puede causar que las regiones detectadas como salientes varíen enormemente de frame a frame. Por el contrario, presentando a la entrada del MLP los valores originales de estas magnitudes, el sistema se vuelve más robusto ante pequeños errores en la estimación de movimiento y la salida es mucho más estable (a, c y e).



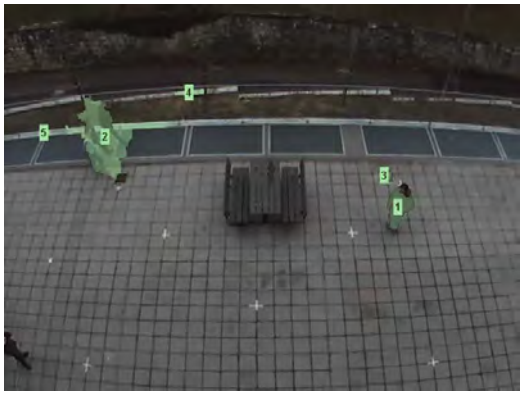
(a) Frame 80, sin normalización



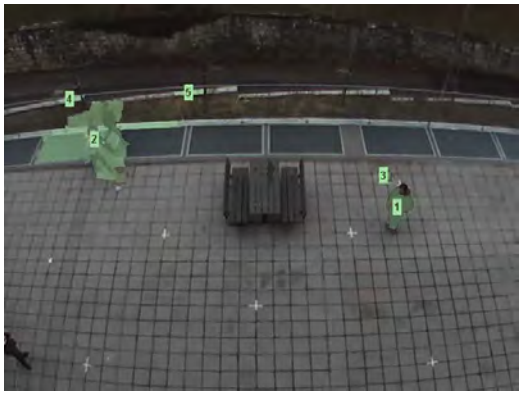
(b) Frame 80, normalizando



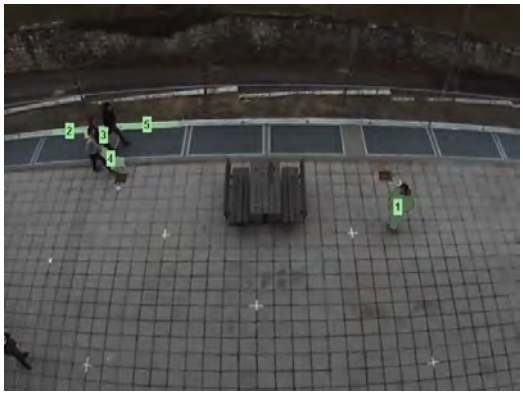
(c) Frame 81, sin normalización



(d) Frame 81, normalizando



(e) Frame 82, sin normalización



(f) Frame 82, normalizando

Figura 3.14. Resultados en '*vigilancia2*' según la normalización del movimiento. Por los mismos motivos que los expuestos en la figura 3.13, podemos observar que la estimación de regiones salientes es mucho más estable cuando no se normaliza el movimiento.

Por último, hemos experimentado también con las diferentes bases de datos que obtuvimos en la sección 3.1. Para poder analizar los resultados, hemos incluido las figuras 3.15 y 3.16, que muestran, respectivamente, datos estadísticos sobre el entrenamiento del perceptrón y la salida que producen en dos fotogramas distintos del vídeo '*vigilancia2*'.

Comenzaremos comentando las curvas ROC y las matrices de confusión expuestas en la figura 3.15. Pese a que la escala es algo pequeña, podemos notar que la curva ROC al utilizar la base de datos 'GT original' (3.15a) es mucho peor que las demás. Como ya discutimos en la sección relativa a la obtención del *ground truth* (sec. 3.1), el método original descrito en el diagrama de la figura 3.4 introducía muchos *outliers*. Ahora podemos confirmar este hecho con datos estadísticos.

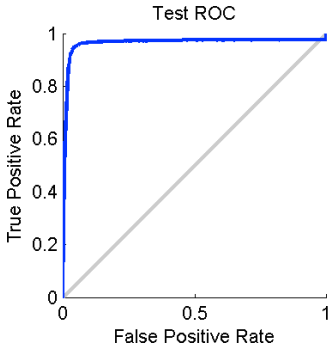
Asimismo, como también habíamos predicho, al entrenar al perceptrón con una cantidad tan alta de valores negativos y en ocasiones contradictorios, la red neuronal ha optado por determinar que ningún segmento es saliente para maximizar su tasa de acierto (3.15b). De hecho, se puede confirmar con las imágenes 3.16a y 3.16b que los resultados empleando 'GT original' son casi aleatorios.

Respecto a las otras tres bases de datos[3] incluidas en la figura 3.15, podemos extraer varias conclusiones. Como podemos ver, la principal ventaja de equilibrar la proporción entre datos positivos y negativos -subfiguras (f) y (h) frente a (d)- es que la red neuronal reduce enormemente la cantidad de falsos negativos: con las bases de datos 'GT 50-50' y 'GT 25-75' apenas se dan casos de segmentos que deberían ser salientes y se clasifican erróneamente.

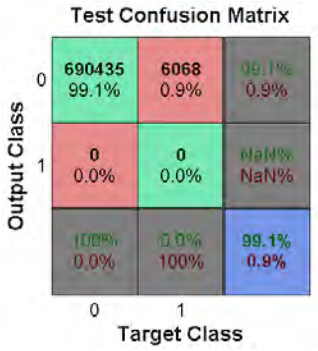
Además, según eliminamos muestras negativas, el efecto producido es que aumenta el número de falsos positivos (zonas teóricamente no salientes se estima que son llamativas). Para observar este hecho de forma más práctica, hemos incluido la figura 3.16. Como podemos ver en ella, la principal diferencia entre las bases de datos 'GT procesado', 'GT 50-50' y 'GT 25-75' reside en el área de las regiones detectadas como salientes.

Por un lado, 'GT procesado' (imágenes c y d) a menudo se queda corto y no detecta completamente a las personas que van en grupo por el centro de la escena. Por el otro, 'GT 50-50' (e y f) suele estimar regiones muy extensas. En el término medio, 'GT 25-75' (g y h) combina lo mejor de cada uno: es capaz de detectar perfectamente a todas las personas, pero sin necesidad de marcar como llamativas áreas tan grandes.

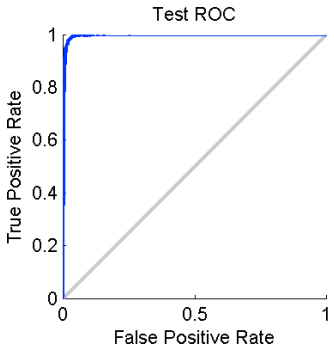
[3] 'GT 10-90' no ha sido incluida porque sus resultados son prácticamente idénticos a 'GT procesado'.



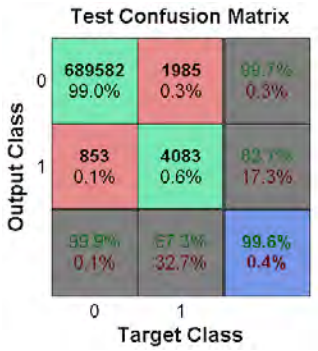
(a) 'GT original'



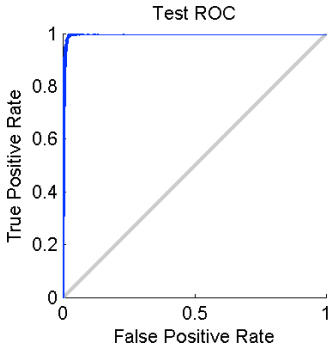
(b) 'GT original'



(c) 'GT procesado'

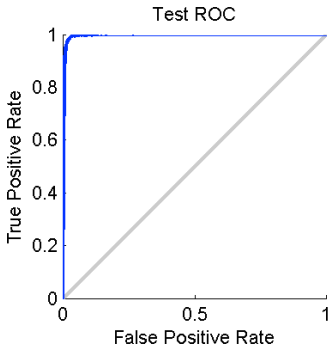


(d) 'GT procesado'

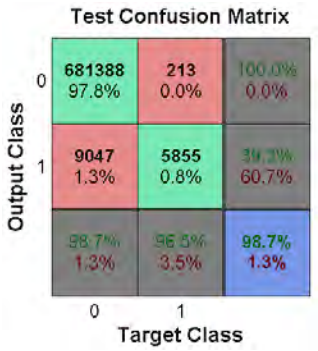


(e) 'GT 50-50'

(f) 'GT 50-50'

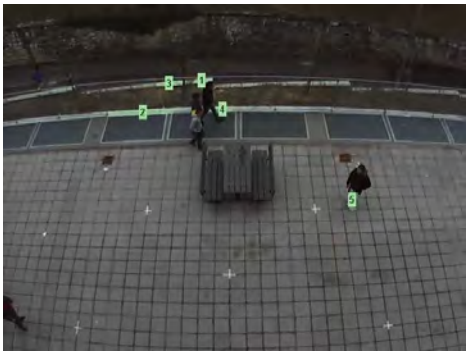


(g) 'GT 25-75'

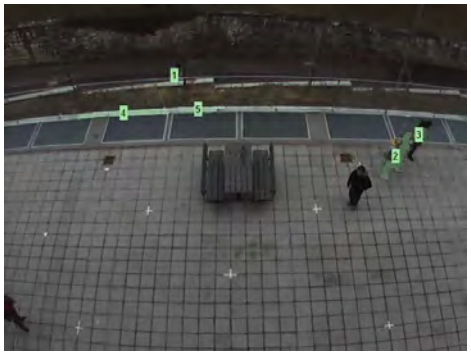


(h) 'GT 25-75'

Figura 3.15. Resultados del entrenamiento según la base de datos utilizada.

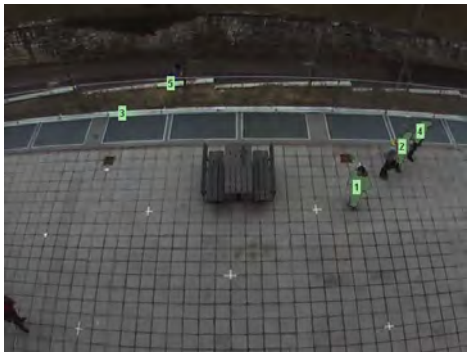


(a) Frame 130, 'GT original'

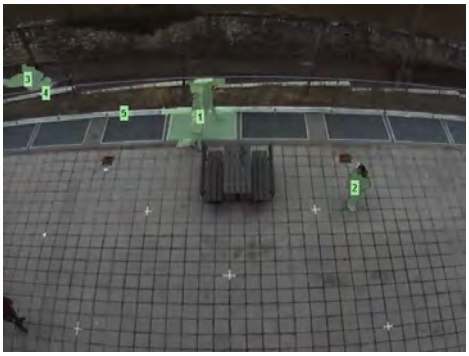


(b) Frame 223, 'GT original'

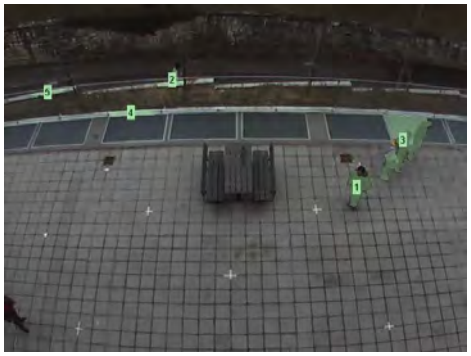
(c) Frame 130, 'GT procesado'



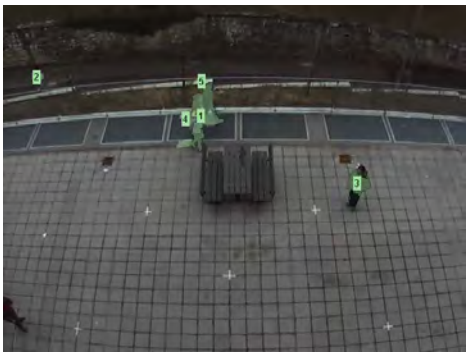
(d) Frame 223, 'GT procesado'



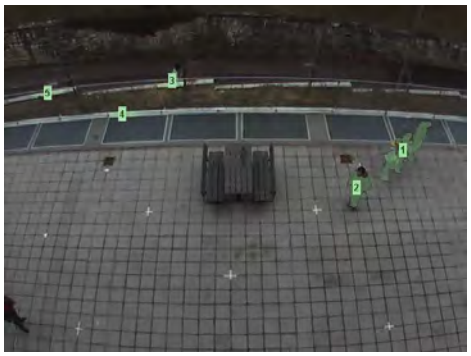
(e) Frame 130, 'GT 50-50'



(f) Frame 223, 'GT 50-50'



(g) Frame 130, 'GT 25-75'



(h) Frame 223, 'GT 25-75'

Figura 3.16. Resultados en '*vigilancia2*' según la base de datos de entrenamiento.

3.3. Resultados

Para terminar este capítulo de experimentos, hemos querido hacer un breve resumen de las conclusiones extraídas tras las diferentes pruebas entrenando a la red neuronal. Con eso, vamos a seleccionar uno de los MLPs y mostrar varios fotogramas del resultado de la detección de saliencia en los vídeos '*vigilancia1*' y '*vigilancia2*'.

Después de analizar el efecto de la variación de los distintos parámetros de los que hemos hablado en la sección anterior, podemos sacar estas tres breves conclusiones:

- Utilizando pocas neuronas en la capa oculta de un MLP, la red neuronal no tiene potencia suficiente para estimar la saliencia en casos complicados. Con nuestro *ground truth*, hemos determinado que los mejores resultados se obtienen con 20 neuronas.

- Normalizar la velocidad y la aceleración en función de su valor máximo en cada fotograma produce variaciones muy rápidas en las regiones que se detectan como las más llamativas. Por ese motivo, preferimos no normalizar el movimiento para obtener resultados más estables.

- Entrenar al modelo con una base de datos muy sesgada hacia valores negativos consigue que la red neuronal no detecte todos los casos positivos que debería. Equilibrando la proporción entre salidas positivas y negativas en la base de datos se logra detectar casi la totalidad de los casos positivos. Tras varias pruebas, hemos obtenido los mejores resultados utilizando una proporción de 1:4.

Por tanto, para analizar los resultados y comparar la eficacia de nuestro modelo frente al `SaliencyToolBox`, hemos utilizado un MLP con 20 neuronas en su capa oculta que hemos entrenado con la base de datos 'GT 25-75' y sin normalizar ni la velocidad ni la aceleración.

Como podemos apreciar en las figuras 3.18 y 3.19, el vídeo con el que fue entrenado el perceptrón ('*vigilancia1_1*') es una secuencia sencilla, con sólo dos personas en escena y poco movimiento. Por este motivo, los resultados de nuestro modelo y el del `SaliencyToolBox` son muy similares. Aún así, podemos ver cómo nuestro algoritmo siempre detecta a las dos personas que aparecen, incluso si están en el borde de la imagen como en la figura 3.19.

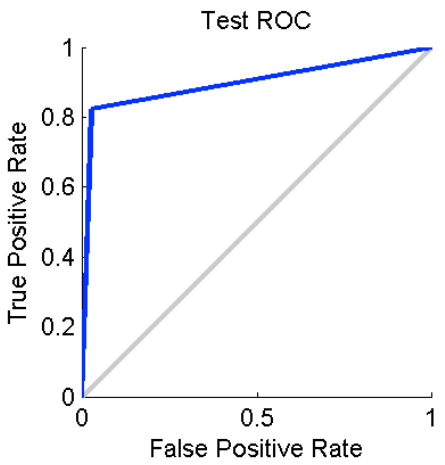
Por el contrario, el vídeo '*vigilancia2*' (figuras 3.20 y 3.21), a pesar de situarse en el mismo escenario, es más complejo. Con varias personas moviéndose por distintas zonas de la escena, podemos observar cómo realmente nuestro modelo es capaz de detectar casi a la perfección todas las regiones de interés, mientras que el `SaliencyToolBox`, debido a la ausencia del factor del movimiento, solo consigue estimar la saliencia de la persona de blanco que se encuentra parada en el centro de la imagen.

Además, como se desprende de la figura 3.17, nuestro modelo claramente supera al `SaliencyToolBox` tanto en precisión como en la curva ROC. Para generar las estadísticas, hemos realizado varios experimentos con el `SaliencyToolBox`. Como esta herramienta produce un mapa de saliencia en escala de grises, hemos necesitado establecer un valor umbral a partir del cual convertir el mapa en una salida binaria.

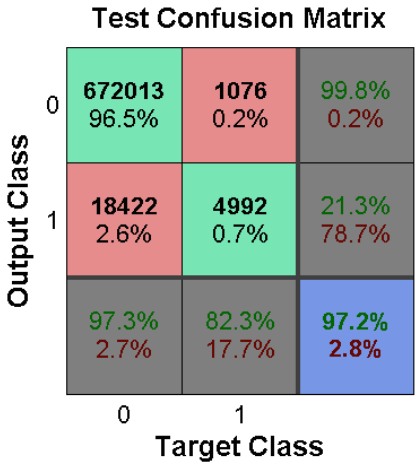
Hemos observado que los mejores resultados se obtienen al fijar dicho valor en 0,3 (sobre 1), como se muestra en las figuras 3.17c y 3.17d, en las que se llega a obtener una tasa de acierto global del 98,8 %. Para añadir también una muestra de otro umbral que consiga una alta tasa de acierto en casos positivos, hemos incluido las figuras 3.17a y 3.17b, en las que el *threshold* se ha establecido en 0,1.

En resumen, podemos confirmar que el modelo que hemos propuesto para calcular la saliencia espacio-temporal supera a uno de los mejores algoritmos desarrollados hasta la fecha para estimar la saliencia estática y produce resultados que se aproximan bastante a la realidad.

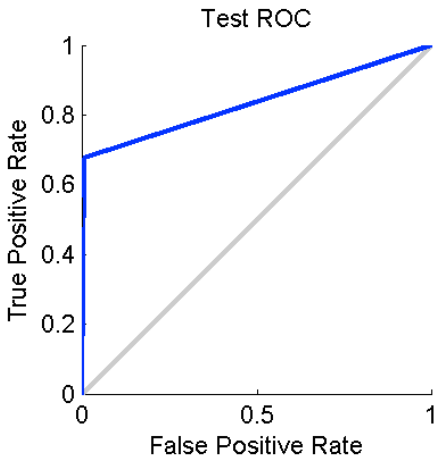
Por último, queremos recordar que, como se explica en el apéndice A, todo el material audiovisual utilizado para este proyecto así como los resultados obtenidos será publicado en el enlace: `https://dl.dropboxusercontent.com/u/17201830/TFG/index.html`.



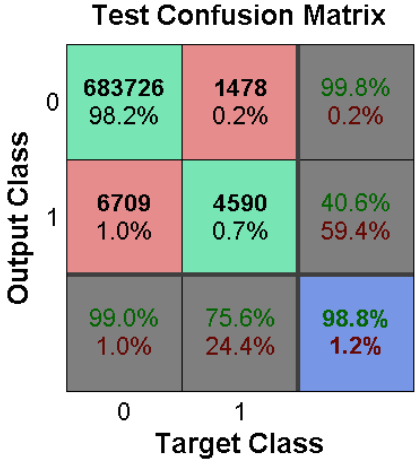
(a) `SaliencyToolBox` > 0,1



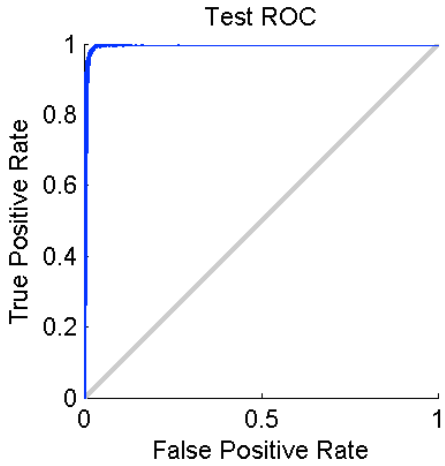
(b) `SaliencyToolBox` > 0,1



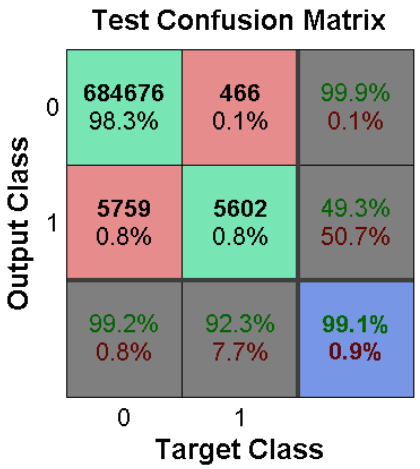
(c) `SaliencyToolBox` > 0,3



(d) `SaliencyToolBox` > 0,3

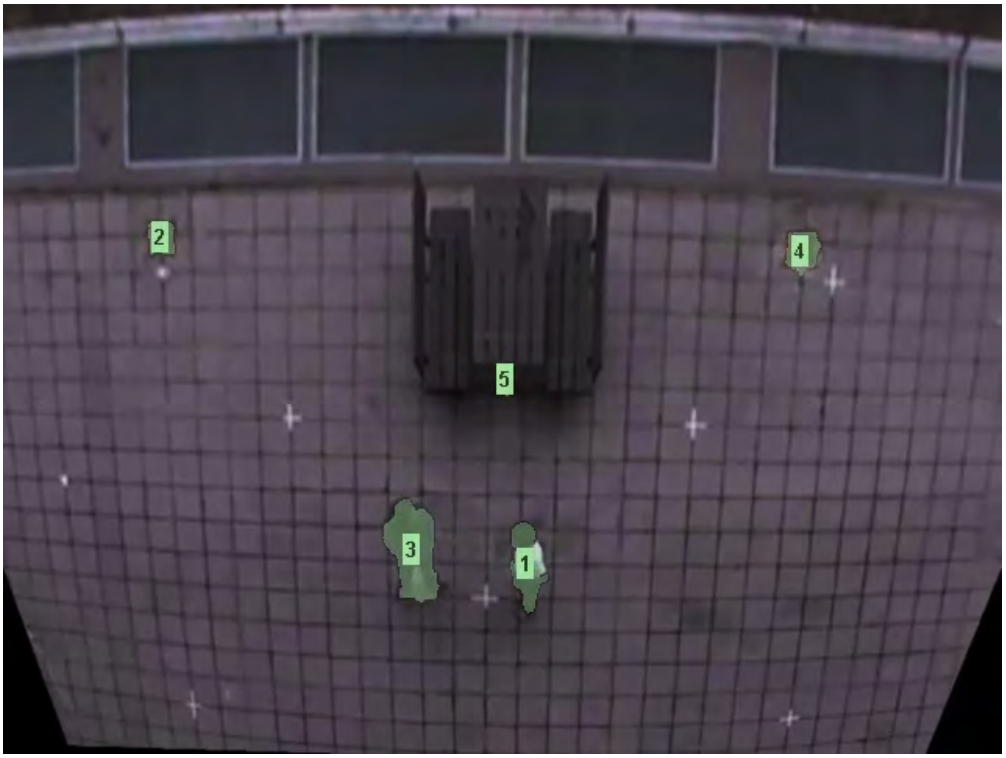


(e) Nuestro modelo (con 'GT 25-75')

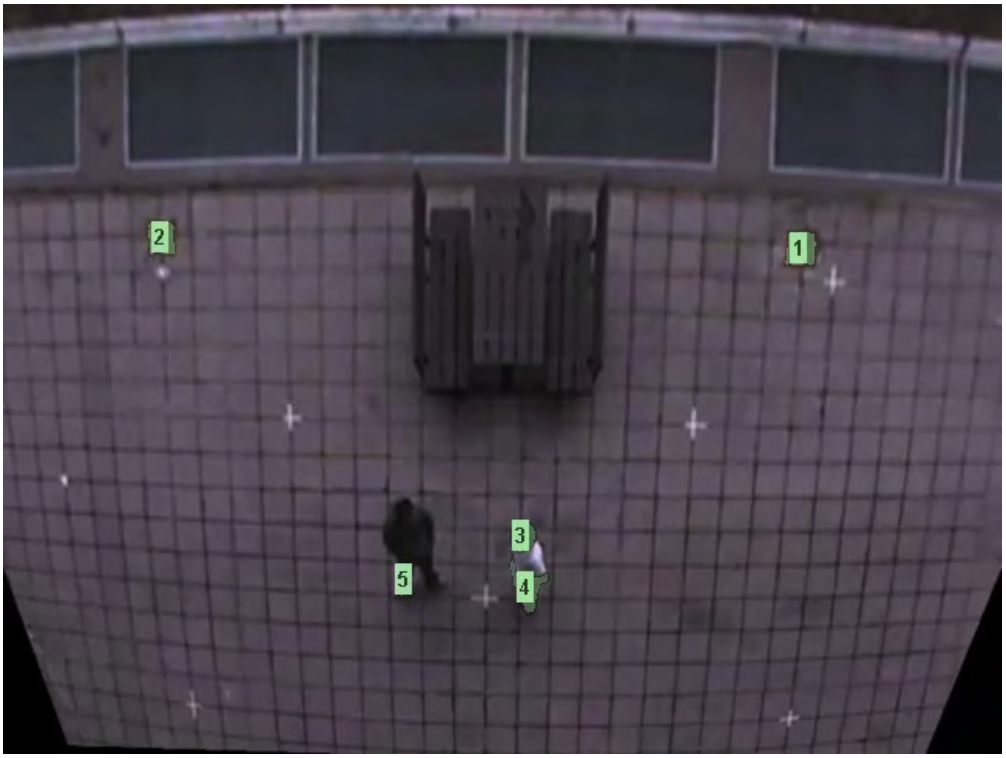


(f) Nuestro modelo (con 'GT 25-75')

Figura 3.17. Comparación de eficacia entre nuestro modelo y el `SaliencyToolBox`.

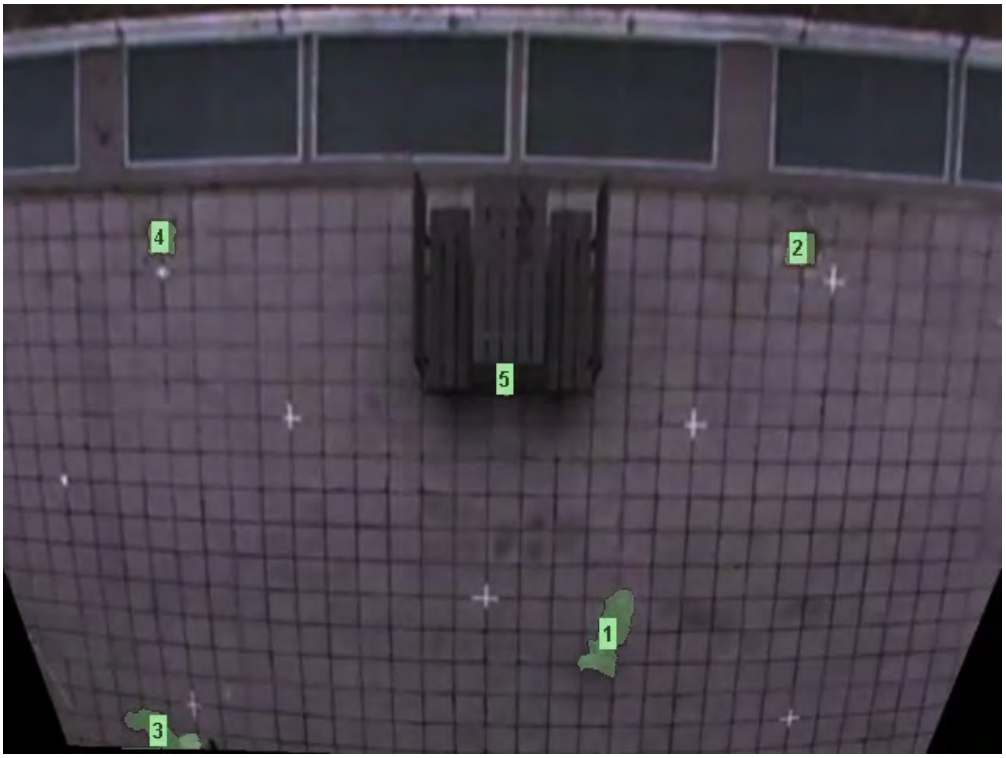


(a) Nuestro modelo

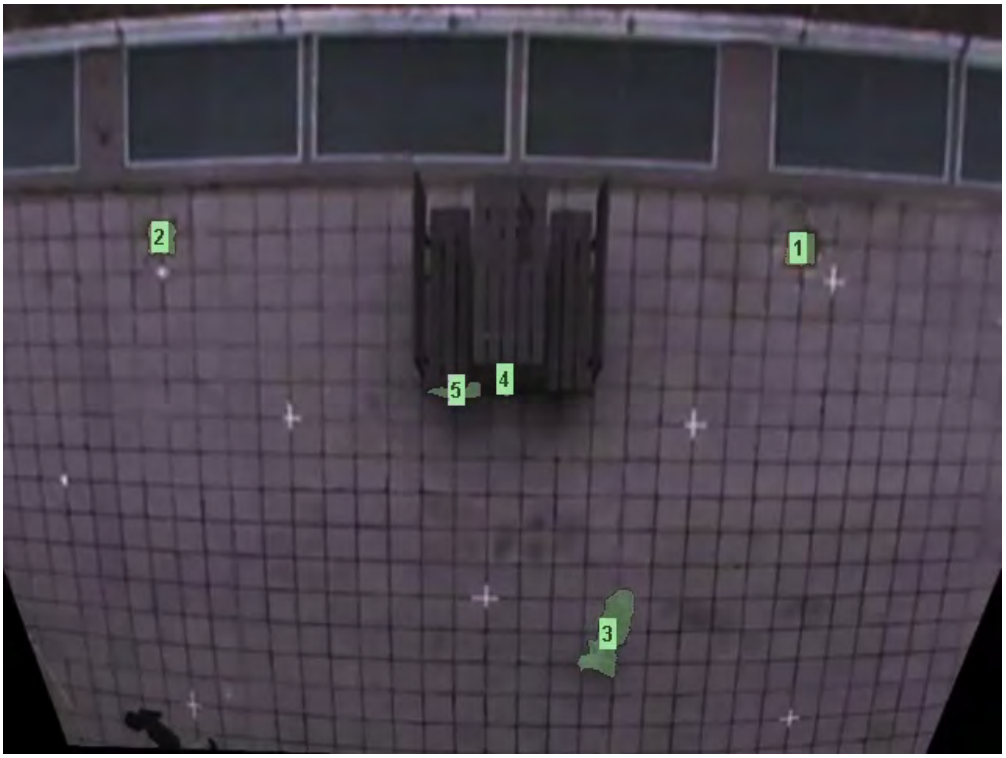


(b) `SaliencyToolBox`

Figura 3.18. Resultados para el frame 25 de '*vigilancia1_2*'.

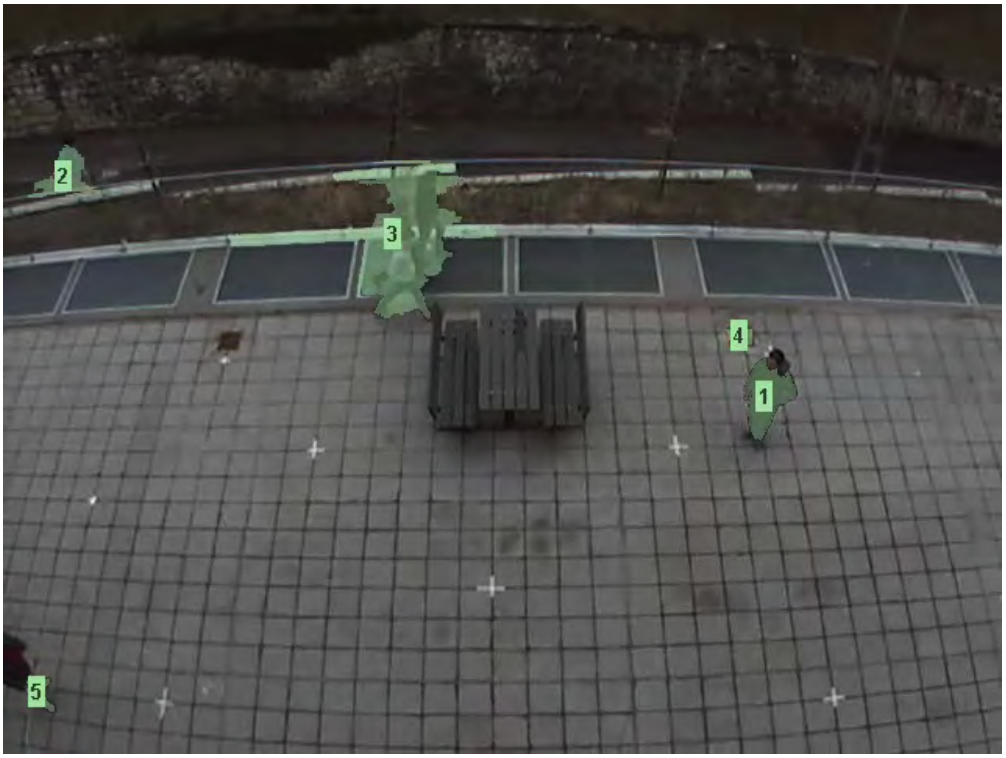


(a) Nuestro modelo

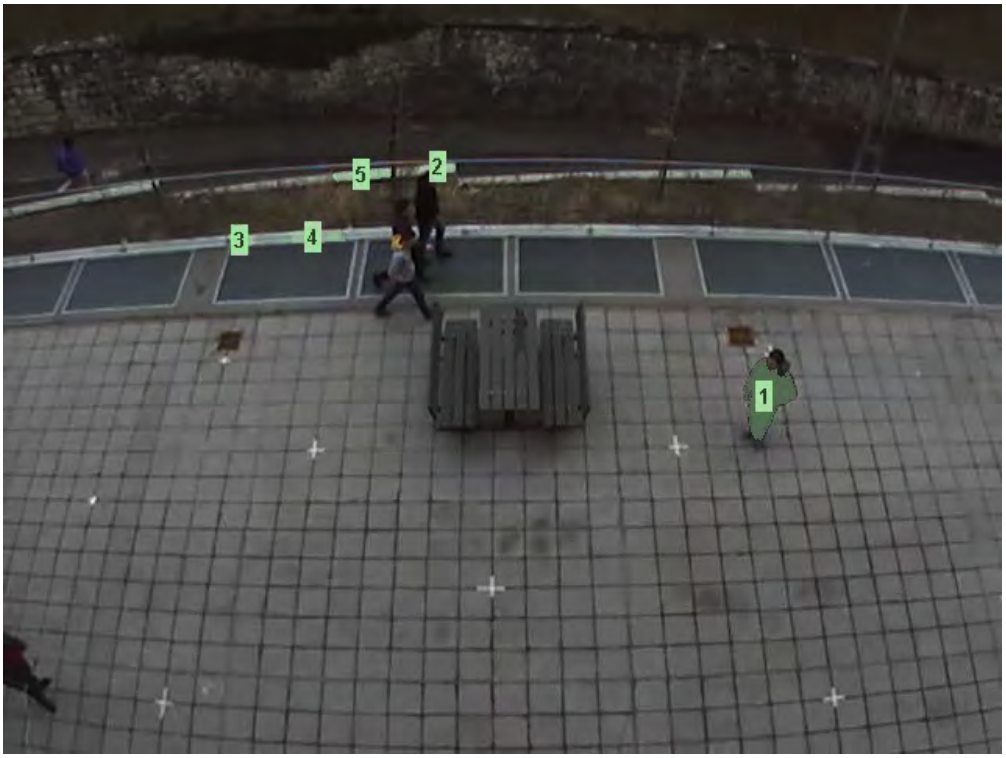


(b) `SaliencyToolBox`

Figura 3.19. Resultados para el frame 2220 de '*vigilancia1_2*'.

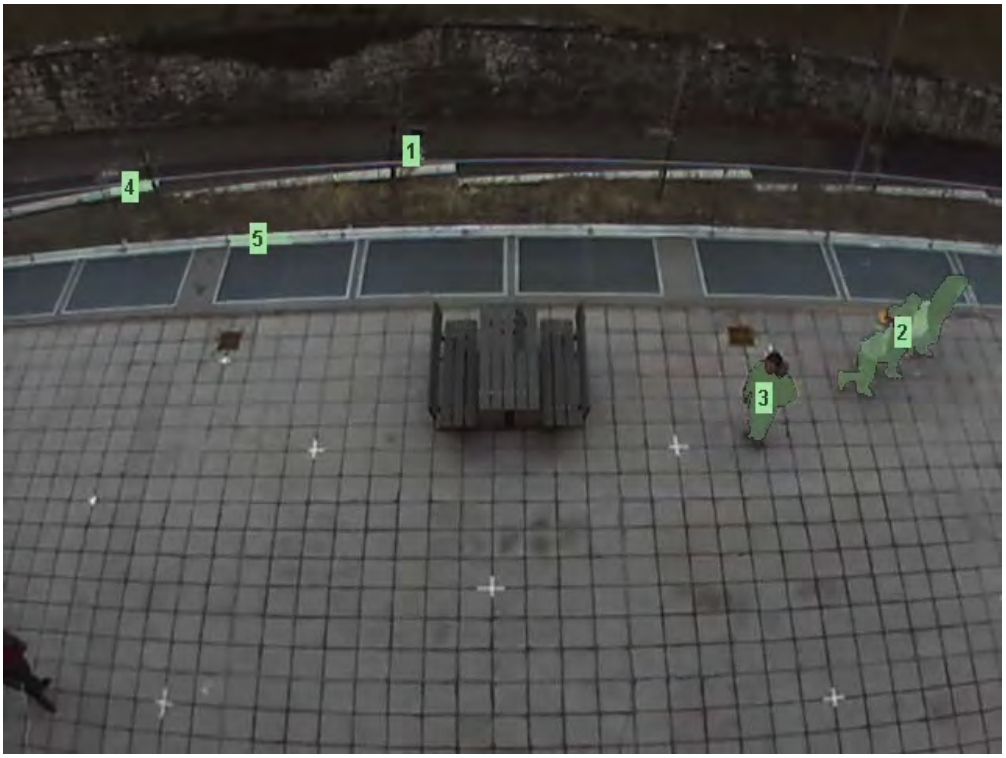


(a) Nuestro modelo

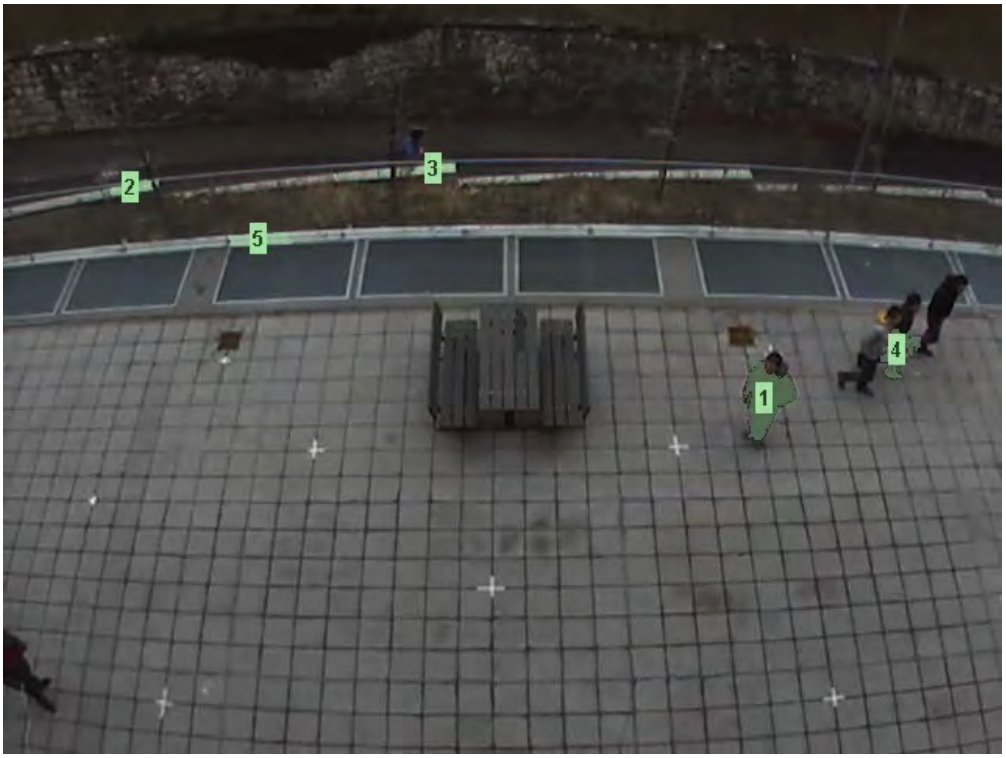


(b) `SaliencyToolBox`

Figura 3.20. Resultados para el frame 127 de '*vigilancia2*'.



(a) Nuestro modelo



(b) `SaliencyToolBox`

Figura 3.21. Resultados para el frame 234 de '*vigilancia2*'.

Capítulo 4
Conclusiones

Tras describir nuestro modelo para la detección de saliencia espacio-temporal y detallar los experimentos realizados para que una red neuronal combine los distintos mapas de características de la manera más óptima, en este último capítulo vamos a exponer las conclusiones de nuestro proyecto.

Además, hemos querido destacar la importancia de nuestra investigación para todo tipo de aplicaciones, desde la detección de objetos hasta la mejora de algoritmos de compresión de vídeo.

Por último, comentaremos la inmensidad de puertas que deja abiertas nuestro proyecto y que, por falta de medios y limitaciones de tiempo, no hemos podido incorporar al presente trabajo, pero que pueden dar lugar a investigaciones futuras muy interesantes y provechosas.

4.1. Conclusiones finales

Como pudimos ver en el primer capítulo, la detección de las regiones más llamativas para el sistema de atención visual humano es un tema muy candente en la comunidad científica especializada en visión artificial desde hace varias décadas.

Desde principios de siglo, los algoritmos propuestos para estimar las zonas de una imagen en las que muy probablemente se fijará una persona, cada vez consiguen mejores resultados.

Ahora que los niveles de acierto de estos algoritmos se encuentran por encima del 90 %, hemos querido ir más allá y proponer un modelo que combine información de diversas fuentes para extender estas potentes aproximaciones en imágenes estáticas a secuencias de vídeo.

Como explicamos en la motivación de este proyecto, nuestra hipótesis es que la única característica que le falta a los modelos de saliencia en fotografías para que produzcan buenos resultados también en secuencias de vídeo es el factor del movimiento.

Para demostrar la validez de nuestra creencia, hemos escogido uno de los mejores algoritmos para la detección de saliencia estática -el `SaliencyToolBox`[64]- y hemos propuesto un modelo para mejorar sus resultados cuando se aplique a vídeos.

Como ya detallamos en el capítulo 2, nuestro modelo incorpora, además del mapa de saliencia que produce el `SaliencyToolBox`, información sobre la magnitud de la velocidad y la aceleración de los objetos, obtenida a partir de los resultados del flujo óptico[66].

Posteriormente, para estimar la importancia de cada una de estas tres características de cara al cálculo del mapa final de saliencia, hemos optado por entrenar a una red neuronal a partir de una pequeña base de datos que hemos conseguido elaborar.

Lamentablemente, debido a que la investigación en saliencia dinámica es muy reciente, aún no existen bases de datos públicas que contengan *ground truth* sobre las regiones más llamativas de algún vídeo.

Gracias a los archivos que utilizó Muratov[1] para su tesis doctoral a los que hemos podido tener acceso, hemos logrado obtener información suficiente para entrenar a la red neuronal.

Sin embargo, al sólo disponer de datos válidos para un único vídeo, no tenemos forma empírica de demostrar que nuestra hipótesis también es válida para otras secuencias distintas de la que se usó para entrenar a nuestro modelo.

Por este motivo, como una de las líneas futuras detalladas en la sección 4.3, hemos planteado realizar numerosos experimentos con un *eye-tracker* para elaborar una gran base de datos que nos ayude a entender cómo influyen el movimiento, la intensidad, la tonalidad o la orientación de los objetos a la hora de que nuestro sistema visual les preste más o menos atención, y poder así proponer mejores modelos que puedan ser aplicados a secuencias reales.

No obstante, basándonos en el *ground truth* que hemos elaborado, hemos conseguido demostrar la validez de nuestra hipótesis: combinando información sobre el movimiento con la salida que produce uno de los mejores algoritmos de saliencia estática, podemos crear un modelo espacio-temporal que supere la eficacia de su predecesor en secuencias de vídeo.

Como muestra la figura 3.17, nuestro algoritmo es capaz de alcanzar tasas de acierto del 99,1 %, mientras que el `SaliencyToolBox` no pasa de 98,8 % en vídeos sencillos con poco movimiento. Por último, queremos destacar que el método que hemos desarrollado también obtiene buenos resultados en otras secuencias con las que no ha sido entrenado, como demuestran las figuras 3.20 y 3.21.

4.2. Aplicaciones de la detección de saliencia

Para destacar la importancia de investigaciones en la línea de nuestro proyecto, hemos querido añadir una sección en la que se muestren ejemplos reales del uso que se está dando actualmente a los escasos algoritmos de saliencia espacio-temporal que se han propuesto hasta ahora.

Por ello, profundizar en estos modelos y lograr mejorar sus resultados puede ser vital para la aparición de nuevas tecnologías en un futuro cercano o el desarrollo de aplicaciones ya existentes para que incorporen nueva funcionalidad.

Así pues, veamos algunos ejemplos en los que se usa o se podría usar un modelo de detección de saliencia espacio-temporal como el nuestro.

Una de las áreas de investigación más importantes en temas de visión es el reconocimiento de objetos. Sin embargo, para poder reconocer un objeto, es evidente que primero hay que detectarlo. Es decir, de entre todos los píxeles de una imagen, el algoritmo tiene que saber de antemano cuáles de ellos forman parte del objeto a reconocer y cuáles no.

Para esta tarea, una simple segmentación no suele ser suficiente, ya que objetos que estén formados por varias partes podrían ser considerados como múltiples elementos distintos. Sin embargo, a partir de la información de saliencia estática y movimiento de los objetos, detectar regiones conectadas en un vídeo puede ser muy sencillo.

Además, si consiguiésemos calcular la saliencia dinámica de una manera más rápida y eficiente, podría servir de gran ayuda para máquinas que tienen recursos limitados, como pueda ser un robot en Marte. En ese caso, un algoritmo de este tipo podría indicarle cuáles son las regiones de interés, de forma que centrase toda su capacidad de procesamiento en esas zonas, sin perder ni tiempo ni recursos en procesar información irrelevante.

Otro ejemplo en el que un modelo de saliencia espacio-temporal podría ser beneficioso consistiría en emplear nuestro algoritmo para que las cámaras digitales mejorasen su función de *autofocus* en la grabación de vídeos. La mayoría, enfocan

automáticamente al empezar a grabar, pero les cuesta adaptar el foco cuando el centro de atención cambia. Siendo capaces de detectar las regiones más salientes a la vez que se graba, podríamos conseguir obtener vídeos que no se desenfoquen en ningún momento.

Por otro lado, modelos como el nuestro también podrían ser usados para mejorar los algoritmos de compresión de vídeo. Durante mucho tiempo se ha investigado en cómo lograr mejores compresiones. Utilizando técnicas de *block-matching*, muchos algoritmos consiguen reducir considerablemente la cantidad de información almacenada para representar una secuencia.

A este proceso, se le podría añadir una fase de detección de las regiones salientes, de forma que se permitiese una reducción de calidad en zonas poco llamativas, manteniendo la máxima posible en las áreas salientes. Gracias a esto, podríamos reducir aún más el tamaño de los archivos de vídeo.

Para demostrar la gran variedad de aplicaciones que puede tener este nuevo campo, queremos recordar el algoritmo de clasificación de vídeos de Rapantzikos et al.[23] (figura 1.7), que se basa en extraer características de la saliencia de los *voxels* a lo largo del vídeo para mejorar notablemente el porcentaje de aciertos con respecto a algoritmos similares que emplean otras características para la clasificación.

También en la sección 1.2 analizamos una curiosa utilidad de la saliencia dinámica: el *video retargeting*, una técnica que consiste en reducir la resolución -y en ocasiones la relación de aspecto- de un vídeo tratando de mantener en todo momento las partes más importantes dentro de la escena.

Anteriormente, esto se había implementado usando *seam carving* (eliminando iterativamente los *rayos* verticales con mínima energía o contraste con respecto a sus vecinos). Gracias a la investigación de Lu et al.[24], la saliencia dinámica puede ser utilizada para determinar con mayor exactitud las zonas importantes que mantener encuadradas (figura 1.8).

Otra de las aplicaciones que vimos en el capítulo del estado del arte fue la descripción de escenas propuesta por Goferman et al.[13]. Como se puede observar en la figura 1.4, su algoritmo emplea la saliencia para determinar la región que mejor describe la imagen. Creemos que este tipo de ideas pueden ser muy interesantes para desarrollar modelos que consigan interpretar el contenido semántico de una fotografía o un vídeo.

Para terminar de explicar la gran utilidad que tiene nuestro proyecto, queremos mencionar una última aplicación de la saliencia espacio-temporal. Actualmente

estamos siendo testigos de una gran revolución tecnológica que, entre la generalización del uso de *smartphones* con cámaras y el gran desarrollo de servicios de almacenamiento como Facebook o YouTube, está dando lugar a la generación de inmensas cantidades de datos visuales en el mundo.

Con tanto contenido, consideramos que un algoritmo que fuese capaz de generar automáticamente un resumen de un vídeo -por ejemplo, mostrando los fotogramas más salientes- podría ser de enorme utilidad.

4.3. Líneas futuras de investigación

Como hemos podido comprobar, el desarrollo de nuestro modelo nos ha abierto una infinidad de puertas con unas posibles aplicaciones realmente interesantes. En esta última sección queremos repasar todos aquellos aspectos que, bien por falta de recursos o bien por escasez de tiempo, no hemos podido incorporar al proyecto y que contribuirían a mejorar los resultados de nuestro algoritmo.

En primer lugar, como ya hemos comentado en las conclusiones, una de las acciones de las que más se podría beneficiar nuestro modelo sería de la realización de un gran número de experimentos.

Utilizando un *eye-tracker* de mayor precisión que el que usó Muratov[1] para su tesis, y recopilando datos de numerosos individuos al visionar diferentes tipos de secuencias, se podría construir una excelente base de datos de la que aprender cómo nuestro cerebro combina la información que le llega en forma de distintos estímulos visuales.

Así, podríamos estimar la ponderación más adecuada entre saliencia espacial, velocidad y aceleración para determinar con mayor exactitud qué regiones de un vídeo tienen una probabilidad más alta de llamar la atención.

Otra de las acciones que se podrían emprender en el futuro sería la modificación y ampliación de nuestro modelo. Por ejemplo, podríamos reemplazar la obtención del mapa de saliencia estática del `SaliencyToolBox` por otro algoritmo más eficaz que surja en los próximos años.

Del mismo modo, viendo el gran esfuerzo que se está dedicando a mejorar los algoritmos de estimación de movimiento, podríamos incrementar la precisión de nuestros cálculos de la velocidad y la aceleración gracias al uso de mejores aproximaciones del flujo óptico. Asimismo, como ya comentamos en la sección 2.2.2, podríamos realizar un filtrado temporal para estimar la aceleración a lo largo de varios fotogramas.

Además, podríamos experimentar incluyendo más características en el modelo antes de entrenarlo con el *ground truth*. Por ejemplo, podríamos incorporar al algoritmo datos como cuánto tiempo lleva la misma región siendo detectada como la más saliente o a qué zonas de la imagen aún no hemos prestado atención. Igualmente, sería interesante explorar otros métodos de aprendizaje máquina, con arquitecturas más sofisticadas y algoritmos de entrenamiento más potentes.

Por último, creemos que también sería una buena idea tratar de optimizar las computaciones para reducir el tiempo de ejecución. En este sentido, implementar el modelo en lenguajes más rápidos como `C++` o incluso en hardware con VHDL podría llegar a conseguir la estimación de la saliencia dinámica en tiempo real.

Apéndices

Apéndice A
Vídeos utilizados

En este apéndice hemos querido incluir información sobre los diferentes vídeos que hemos utilizado y sobre cómo hemos dividido la secuencia de la que disponemos de *ground truth* en dos partes para poder entrenar la red neuronal con una y medir los resultados en la otra.

Aunque hemos hecho pruebas con otras secuencias, para este proyecto hemos utilizado únicamente dos vídeos, que hemos denominado '*vigilancia1*' y '*vigilancia2*'. En los cuadros A.1 y A.2 hemos incluido información básica sobre cada uno de ellos. Además, queremos destacar que todo el material audiovisual será colgado en el enlace `dl.dropboxusercontent.com/u/17201830/TFG/index.html`.

Queremos comentar también que en el primer vídeo -'*vigilancia1*'- hemos aplicado un diezmado de factor 2 puesto que los tiempos de ejecución de los algoritmos de saliencia estática y flujo óptico eran demasiado largos. Utilizando una resolución de 640x480, la obtención de los tres mapas de características -saliencia espacial, velocidad y aceleración- para el vídeo completo tardó unos 6 días y 9 horas.

Formato:	MPEG-4
Resolución original:	1 280 x 960 px
Resolución de trabajo:	640 x 480 px
Formato de píxel:	RGB24
Duración:	10 min 22.93 seg
Número de tramas:	18 689 frames
Tasa de trama:	30 fps
Tasa binaria:	3 336 kbps

Cuadro A.1. Datos sobre el vídeo '*vigilancia1*'.

Formato:	MPEG-4
Resolución original:	640 x 480 px
Resolución de trabajo:	640 x 480 px
Formato de píxel:	RGB24
Duración:	12.75 seg
Número de tramas:	256 frames
Tasa de trama:	20 fps
Tasa binaria:	2 093 kbps

Cuadro A.2. Datos sobre el vídeo '*vigilancia2*'.

Como hemos explicado a lo largo del capítulo 3, el único vídeo del que hemos podido obtener datos fiables como *ground truth* ha sido el de '*vigilancia1*'. Para poder utilizar técnicas de aprendizaje máquina y comparar la eficacia de nuestro modelo con la del algoritmo de saliencia estática del `SaliencyToolBox` disponiendo de un único vídeo, hemos necesitado dividir la secuencia completa en dos partes, que hemos denominado '*vigilancia1_1*'y '*vigilancia1_2*'.

Así, hemos utilizado '*vigilancia1_1*'para entrenar a la red neuronal y '*vigilancia1_2*', más corto, para los *test*. Por último, hemos añadido en los cuadros A.3 y A.4 información adicional sobre estas dos secuencias[1]:

Duración:	8 min 20 seg
Número de tramas:	15 000 frames
Tasa binaria:	1 731 kbps

Cuadro A.3. Datos sobre la secuencia '*vigilancia1_1*'.

Duración:	2 min 2.93 seg
Número de tramas:	3 689 frames
Tasa binaria:	1 627 kbps

Cuadro A.4. Datos sobre la secuencia '*vigilancia1_2*'.

[1] Los parámetros que no han sido incluidos en estas tablas conservan el mismo valor que en el vídeo original, '*vigilancia1*' (cuadro A.1).

Apéndice B

Presupuesto

En este apéndice hemos incluido la justificación de los costes totales de la realización de este Trabajo Fin de Grado. Como se puede leer de las tablas B.1 y B.2, estos costes se pueden dividir entre gastos de personal y de material.

En el cuadro B.1 se detallan las fases del proyecto y el tiempo aproximado que hemos empleado en cada una de ellas. Así pues, se desprende que el tiempo total dedicado por el proyectando ha sido de 1 100 horas, de las cuales aproximadamente el 25 % han sido compartidas con la tutora del proyecto, por lo que la cantidad final asciende a 1 375 horas.

Según las tablas de Recursos Humanos para la contratación de personal en la Universidad Carlos III a las que hemos tenido acceso, los honorarios de un profesor titular son de 36,57 €/hora, mientras que las tarifas para estudiantes se establecen en 19,86 €/hora. A partir de estos datos, las ecuaciones B.1, B.2 y B.3 indican cómo obtener el coste total de personal, C_P, siendo H_e, n_e, H_t y n_t los honorarios y número de horas empleadas por estudiante y tutor respectivamente.

Fase	Descripción	Tiempo empleado
1	Documentación	300 horas
2	Diseño del modelo	150 horas
3	Implementación del modelo en software	300 horas
4	Entrenamiento del modelo y análisis de resultados	150 horas
5	Redacción de la memoria del proyecto	200 horas

Cuadro B.1. Fases del proyecto.
Listado de las diferentes fases del proyecto y de las horas dedicadas a cada una de ellas.

Concepto	Importe
Ordenador de gama media	1 500 €
Licencia de estudiante para Matlab	76 €
Documentación	200 €
Gastos varios	300 €

Cuadro B.2. Costes de material.

$$C_{P_e} = H_e \times n_e = 19,86 \times 1100 = 21\,846 \text{ €} \tag{B.1}$$

$$C_{P_t} = H_t \times n_t = 36,57 \times 275 = 10\,056,75 \text{ €} \tag{B.2}$$

$$C_P = C_{P_e} + C_{P_t} = 31\,902,75 \text{ €} \tag{B.3}$$

Por otro lado, en el cuadro B.2 se recogen los costes de material desglosados en: equipo informático, licencia de software ('MATLAB and Simulink Student Suite + Neural Network Toolbox'), acceso a bibliotecas y liberías de documentación y gastos varios no atribuibles (material fungible, llamadas telefónicas, desplazamientos, etcétera). Como se desprende del cuadro B.2, el coste de material se sitúa en los 2 076 €.

A partir de estos datos, hemos construido el cuadro B.3, en el que se calcula el valor total del presupuesto del presente Trabajo Fin de Grado, que asciende finalmente a 41 114,25 €.

Concepto	Importe
Costes de personal	31 902,75 €
Costes de material	2 076 €
Base imponible	33 978,75 €
I.V.A. (21 %)	7 135,5 €
TOTAL	**41 114,25 €**

Cuadro B.3. Presupuesto total del TFG.

Bibliografía

[1] O. Muratov, "Visual Saliency Detection and its Application to Image Retrieval", *Università degli studi di Trento*, 2013, URL `http://eprints-phd.biblio.unitn.it/892/`.

[2] V. Menon y L. Q. Uddin, "Saliency, switching, attention and control: a network model of insula function", *Brain Structure and Function*, 214(5-6):655–667, Mayo de 2010, URL `http://link.springer.com/10.1007/s00429-010-0262-0`.

[3] S. Treue, "Visual attention: the where, what, how and why of saliency", *Current opinion in neurobiology*, 13(4):428–432, 2003, URL `http://www.sciencedirect.com/science/article/pii/S0959438803001053`.

[4] L. Itti, C. Koch y E. Niebur, "A Model of Saliency-Based Visual Attention for Rapid Scene Analysis", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, Noviembre de 1998, URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=730558`.

[5] L. Itti y C. Koch, "A saliency-based search mechanism for overt and covert shifts of visual attention", *Vision research*, 2000, URL `http://www.sciencedirect.com/science/article/pii/S0042698999001637`.

[6] N. Bruce y J. Tsotsos, "Saliency based on information maximization", *Advances in Neural Information Processing Systems*, 2006, URL `https://papers.nips.cc/paper/2830-saliency-based-on-information-maximization.pdf`.

[7] B. Schölkopf, J. Platt y T. Hofmann, "Graph-Based Visual Saliency", en *Advances in Neural Information Processing Systems*, págs. 545–552, 2006, URL `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6287326`.

[8] X. Hou y L. Zhang, "Saliency Detection: A Spectral Residual Approach", en *2007 IEEE Conference on Computer Vision and Pattern Recognition*, págs. 1–8, IEEE, 2007, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4270292.

[9] T. Judd, K. Ehinger, F. Durand y A. Torralba, "Learning to predict where humans look", en *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, págs. 2106–2113, IEEE, 2009, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5459462.

[10] P. Viola y M. J. Jones, "Robust Real-Time Face Detection", *International Journal of Computer Vision*, 57(2):137–154, Mayo de 2004, URL http://link.springer.com/10.1023/B:VISI.0000013087.49260.fb.

[11] F. Perazzi, P. Krahenbuhl, Y. Pritch y A. Hornung, "Saliency filters: Contrast based filtering for salient region detection", en *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, págs. 733–740, IEEE, 2012, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6247743.

[12] M.-M. Cheng, J. Warrell, W.-Y. Lin, S. Zheng, V. Vineet y N. Crook, "Efficient Salient Region Detection with Soft Image Abstraction", en *2013 IEEE International Conference on Computer Vision (ICCV)*, págs. 1529–1536, IEEE, 2013, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6751300.

[13] S. Goferman, L. Zelnik-Manor y A. Tal, "Context-Aware Saliency Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(10):1915–1926, Octubre de 2012, URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6112774.

[14] O. Le Meur, D. Thoreau, P. Le Callet y D. Barba, "A spatio-temporal model of the selective human visual attention", en *International Conference on Image Processing*, págs. III–1188, IEEE, 2005, URL https://www.irisa.fr/temics/publis/2005/LeMeur_ICIP05.pdf.

[15] Y. Zhai y M. Shah, "Visual attention detection in video sequences using spatiotemporal cues", en *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, p. 815, ACM Request Permissions, New York, New York, USA, Octubre de 2006, URL http://portal.acm.org/citation.cfm?doid=1180639.1180824.

[16] J. Shi y Tomasi, "Good features to track", en *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, págs. 593–600, IEEE Comput. Soc. Press, 1994, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=323794.

[17] S. Marat, T. Ho Phuoc, L. Granjon, N. Guyader, D. Pellerin y A. Guérin-Dugué, "Spatio-temporal saliency model to predict eye movements in video free viewing", en *Proceedings of the 16th European Signal Processing Conference, EUSIPCO-2008*, págs. 1–5, Grenoble Images Parole Signal Automatique - GIPSA-lab, Lausanne, Suisse, Junio de 2008, URL http://hal.archives-ouvertes.fr/hal-00288966.

[18] S. Marat, T. H. Phuoc, L. Granjon, N. Guyader, D. Pellerin y A. Guérin-Dugué, "Modelling Spatio-Temporal Saliency to Predict Gaze Direction for Short Videos", *International Journal of Computer Vision*, 82(3):231–243, Mayo de 2009, URL http://link.springer.com/10.1007/s11263-009-0215-3.

[19] C. Liu, P. C. Yuen y G. Qiu, "Object motion detection using information theoretic spatio-temporal saliency", *Pattern Recognition*, 42(11):2897–2906, Noviembre de 2009, URL http://linkinghub.elsevier.com/retrieve/pii/S0031320309000685.

[20] C. Chamaret, J. C. Chevet y O. Le Meur, "Spatio-temporal combination of saliency maps and eye-tracking assessment of different strategies", en *2010 17th IEEE International Conference on Image Processing (ICIP 2010)*, págs. 1077–1080, IEEE, 2010, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5651381.

[21] C. Guo, Q. Ma y L. Zhang, "Spatio-temporal Saliency detection using phase spectrum of quaternion fourier transform", en *2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, págs. 1–8, IEEE, 2008, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4587715.

[22] S. Zhong, Y. Liu, F. Ren, J. Zhang y T. Ren, "Video Saliency Detection via Dynamic Consistent Spatio-Temporal Attention Modelling", *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013, URL http://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/viewFile/6387/7341.

[23] K. Rapantzikos, N. Tsapatsoulis, Y. Avrithis y S. Kollias, "Spatio-temporal saliency for video classification", *Signal Processing: Image Communication*, 24(7):557–571, Agosto de 2009, URL http://linkinghub.elsevier.com/retrieve/pii/S0923596509000320.

[24] T. Lu, Z. Yuan, Y. Huang, D. Wu y H. Yu, "Video retargeting with nonlinear spatial-temporal saliency fusion", en *2010 17th IEEE International Conference on Image Processing (ICIP 2010)*, págs. 1801–1804, IEEE, 2010, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5651644.

[25] D. Nistér, "Preemptive RANSAC for live structure and motion estimation", *Machine Vision and Applications*, 16(5):321–329, Diciembre de 2005, URL http://link.springer.com/10.1007/s00138-005-0006-y.

[26] M. A. Fischler y R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography", *Communications of the ACM*, 24(6):381–395, Junio de 1981, URL http://portal.acm.org/citation.cfm?doid=358669.358692.

[27] A. Hast y J. Nysjö, "Optimal RANSAC - Towards a Repeatable Algorithm for Finding the Optimal Set", *Journal of WSCG*, 21(1):21–30, 2013, URL http://wscg.zcu.cz/wscg2013/program/full/A53-full.pdf.

[28] M. Ghanbari, "The cross-search algorithm for motion estimation", *IEEE Transactions on Communications*, 38(7):950–953, Julio de 1990, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=57512.

[29] P. Anandan, J. R. Bergen y K. J. Hanna, "Hierarchical model-based motion estimation", *Computer Vision - ECCV'92*, 588:237–252, 1992, URL http://link.springer.com/chapter/10.1007/3-540-55426-2_27.

[30] R. Li, B. Zeng y M. L. Liou, "A new three-step search algorithm for block motion estimation", *IEEE Transactions on Circuits and Systems for Video Technology*, 4(4):438–442, 1994, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=313138.

[31] W. Li y E. Salari, "Successive elimination algorithm for motion estimation", *IEEE Transactions on Image Processing*, 4:105–107, Enero de 1995, URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=350809.

[32] L.-M. Po y W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation", *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):313–317, Junio de 1996, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=499840.

[33] J. Y. Tham, S. Ranganath, M. Ranganath y A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation", *IEEE Transactions on Circuits and Systems for Video Technology*, 8(4):369–377, 1998, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=709403.

[34] S. Zhu y K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation", *IEEE Transactions on Image Processing*, 9:287–290, Febrero de 2000, URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=821744.

[35] C. Zhu, X. Lin y L.-P. Chau, "Hexagon-based search pattern for fast block motion estimation", *IEEE Transactions on Circuits and Systems for Video Technology*, 12(5):349–355, Mayo de 2002, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1003474.

[36] B. K. Horn, B. K. P. Horn y B. G. Schunck, "Determining Optical Flow", en *1981 Technical Symposium East*, págs. 319–331, SPIE, Noviembre de 1981, URL http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=1231385.

[37] J. McDermott, Y. Weiss y E. H. Adelson, "Beyond junctions: nonlocal form constraints on motion interpretation", *Perception*, 30(8):905–923, 2001, URL http://www.perceptionweb.com/abstract.cgi?id=p3219.

[38] D. J. Heeger, "Optical flow using spatiotemporal filters", *International Journal of Computer Vision*, 1(4):279–302, 1988, URL http://link.springer.com/10.1007/BF00133568.

[39] A. Verri y T. Poggio, "Motion Field and Optical Flow: Qualitative Properties", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):490–498, Mayo de 1989, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=24781.

[40] E. P. Simoncelli, E. H. Adelson y D. J. Heeger, "Probability distributions of optical flow", *1991 IEEE Computer Society Conference on Com-*

puter Vision and Pattern Recognition, págs. 310–315, 1991, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=139707.

[41] M. Proesmans, L. Van Gool y E. Pauwels, "Determination of optical flow and its discontinuities using non-linear diffusion", *Computer Vision - ECCV'94*, págs. 294–304, 1994, URL http://link.springer.com/chapter/10.1007/BFb0028362.

[42] J. L. Barron, D. J. Fleet y S. S. Beauchemin, "Performance of optical flow techniques", *International Journal of Computer Vision*, 12(1):43–77, Febrero de 1994, URL http://link.springer.com/10.1007/BF01420984.

[43] B. Galvin, B. McCane, K. Novins, D. Mason y S. Mills, "Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms.", *BMVC*, 1998, URL http://ami.dis.ulpgc.es/biblio/bibliography/documentos/galvin98recovering.pdf.

[44] S. Baker, S. Roth, D. Scharstein, M. J. Black, J. P. Lewis y R. Szeliski, "A Database and Evaluation Methodology for Optical Flow", en *2007 IEEE 11th International Conference on Computer Vision*, págs. 1–8, IEEE, 2007, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4408903.

[45] C. Liu, W. T. Freeman, E. H. Adelson y Y. Weiss, "Human-assisted motion annotation", en *2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, págs. 1–8, IEEE, 2008, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4587845.

[46] T. Brox, A. Bruhn, N. Papenberg y J. Weickert, "High accuracy optical flow estimation based on a theory for warping", *Computer Vision - ECCV 2004*, 2004, URL http://link.springer.com/chapter/10.1007/978-3-540-24673-2_3.

[47] A. Bruhn, J. Weickert y C. Schnörr, "Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods", *International Journal of Computer Vision*, 61(3):211–231, 2005, URL http://link.springer.com/10.1023/B:VISI.0000045324.43199.43.

[48] D. Fleet y Y. Weiss, "Optical flow estimation", *Handbook of Mathematical Models in Computer Vision*, págs. 237–257, 2006, URL http://link.springer.com/10.1007/0-387-28831-7_15.

[49] L. Alvarez, R. Deriche, T. Papadopoulo y J. Sánchez, "Symmetrical Dense Optical Flow Estimation with Occlusions Detection", *International Journal of Computer Vision*, 75(3):371–385, Febrero de 2007, URL http://link.springer.com/10.1007/s11263-007-0041-4.

[50] D. Sun, S. Roth y M. J. Black, "Secrets of optical flow estimation and their principles", *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, págs. 2432–2439, 2010, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5539939.

[51] S. Lloyd, "Least squares quantization in PCM", *IEEE Transactions on Information Theory*, 28(2):129–137, Marzo de 1982, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1056489.

[52] J. A. Hartigan y M. A. Wong, "Algorithm AS 136: A K-Means Clustering Algorithm", *Applied Statistics*, 28(1):100–108, 1979, URL http://www.jstor.org/stable/10.2307/2346830?origin=crossref.

[53] C. M. Christoudias, B. Georgescu y P. Meer, "Synergism in low level vision", en *16th International Conference on Pattern Recognition*, págs. 150–155, IEEE Comput. Soc, 2002, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1047421.

[54] D. Comaniciu y P. Meer, "Mean shift: a robust approach toward feature space analysis", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, Mayo de 2002, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1000236.

[55] P. Meer y B. Georgescu, "Edge detection with embedded confidence", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1351–1365, 2001, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=977560.

[56] M. Grundmann, V. Kwatra, M. Han y I. Essa, "Efficient hierarchical graph-based video segmentation", en *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, págs. 2141–2148, IEEE, 2010, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5539893.

[57] D. E. Rumelhart, G. E. Hinton y R. J. Williams, "Learning internal representations by error propagation", *Readings in Cognitive Science*, 1985,

URL http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA164453.

[58] D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters", *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, Junio de 1963, URL http://epubs.siam.org/doi/abs/10.1137/0111030.

[59] M. T. Hagan y M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm", *IEEE Transactions on Neural Networks*, 5(6):989–993, 1994, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=329697.

[60] D. J. C. MacKay, "Bayesian Interpolation", *Neural Computation*, 4(3):415–447, Mayo de 1992, URL http://www.mitpressjournals.org/doi/abs/10.1162/neco.1992.4.3.415.

[61] F. Dan Foresee y M. T. Hagan, "Gauss-Newton approximation to Bayesian learning", en *International Conference on Neural Networks (ICNN'97)*, págs. 1930–1935, IEEE, 1997, URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=614194.

[62] M. Riedmiller y H. Braun, "A direct adaptive method for faster backpropagation learning: the RPROP algorithm", en *IEEE International Conference on Neural Networks*, págs. 586–591, IEEE, 1993, URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=298623.

[63] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning", *Neural Networks*, 6(4):525–533, 1993, URL http://linkinghub.elsevier.com/retrieve/pii/S0893608005800565.

[64] D. Walther y C. Koch, "Modeling attention to salient proto-objects", *Neural Networks*, 19(9):1395–1407, Noviembre de 2006, URL http://linkinghub.elsevier.com/retrieve/pii/S0893608006002152.

[65] D. Walther, "Interactions of visual attention and object recognition: computational modeling, algorithms, and psychophysics", *California Institute of Technology*, 2006, URL http://thesis.library.caltech.edu/895/1/00_DirkWalther_PhDthesis.pdf.

[66] C. Liu, "Beyond pixels: exploring new representations and applications for motion analysis", *Massachusetts Institute of Technology*, 2009, URL `http://dspace.mit.edu/handle/1721.1/53293`.

[67] A. Armanini y N. Conci, "Eye tracking as an accessible assistive tool", *2010 11th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, págs. 1–4, 2010, URL `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5617660`.

Índice de Figuras

Índice de Cuadros